

Aalto University
School of Science and Technology
Degree Programme of Computer Science and Engineering

Mikael Lavi

Implementing Touch-based User Interface for Existing Software

Aalto-yliopisto
Perustieteiden korkeakoulu
Tietotekniikan talon kirjasto

Master's Thesis
Espoo, May 15, 2012

Supervisor:
Instructor:

Professor Lauri Savioja
Marko Myllymaa, M.Sc.

Aalto University School of Science Degree Programme in Computer Science and Engineering		ABSTRACT OF THE MASTER'S THESIS	
Author: Mikael Lavi			
Title: Implementing Touch-based User Interface for Existing Software			
Number of pages: 99	Date: 15.5.2012	Language: English	
Professorship: Media technology		Code: T-111	
Supervisor: Professor Lauri Savioja			
Instructor(s): Marko Myllymaa, M.Sc.			
<p>Abstract:</p> <p>The purpose of this work was to evaluate the migration steps of a windowing desktop application into a touch based input enabled software.</p> <p>The study was conducted on an already existing building information modeling software called Tekla BIMsight. The task was to retain all the functionality already in the software while making it possible to be used on touch-enabled devices, such as tablets or convertible laptops with a swivel display. Design and implementation of the system has been documented as part of the thesis, as well as most problematic issues during this period. The effects of the implementation are validated and tested with real users and the results from that study were documented. The usability study was conducted to obtain quantitative and qualitative metrics of the usability.</p> <p>The nature of the input mechanism, direct or indirect, affects the user experience greatly. The final system should be as responsive as possible to maintain a good level of perceived performance. Early prototyping and access to the target devices is critical to the success of a migration process. There are several common mistakes that should be avoided in the design and implementation phases. Not all the problems were critical, but many of them were identified as very cumbersome for the user that would affect the positive user experience of the software. With each new context for a user interface the problems need to be solved again and only experience from such solutions can help alleviate this task.</p> <p>The implemented touch support can be verified to meet the set requirements very well: It allows the system to be used on touch based input environments and all the major user interface elements support this.</p>			
Keywords: HCI, BIM, UI, touch, user interface, tablet, review software, evaluation			

Aalto-yliopisto Perustieteiden korkeakoulu Tietotekniikan koulutusohjelma		DIPLOMITYÖN TIIVISTELMÄ	
Tekijä: Mikael Lavi			
Työn nimi: Kosketuskäyttöliittymän toteuttaminen olemassa olevaan ohjelmaan			
Sivumäärä: 99	Päiväys: 15.5.2012	Julkaisukieli: Englanti	
Professori: Mediatekniikka		Professuurikoodi: T-111	
Työn valvoja: Professori Lauri Savioja			
Työn ohjaaja(t): Marko Myllymaa, DI			
<p>Tiivistelmä:</p> <p>Työn tarkoituksena oli toteuttaa ja arvioida toimenpiteet ja menetelmät joilla olemassa olevaan käyttöliittymään voidaan lisätä tuki kosketuskäytölle.</p> <p>Ominaisuudet lisättiin rakennusten tietomallinnuksen tarkasteluohjelmaan, Tekla BIMsight. Tehtävänä oli säilyttää kaikki aiemmat toiminnot ja tehdä ohjelmasta tehokkaasti käytettävä kosketuslaitteilla, kuten tableteilla ja kääntyvällä näytöllä varustetuilla kannettavilla. Suunnittelu ja toteutus järjestelmälle on dokumentoitu työssä ja kaikkein vaativimmat ongelmat. Toteutetun tuen vaikutuksia arvioitiin oikeiden käyttäjien kanssa tehdyssä käyttäjätutkimuksessa, jonka tulokset on esitetty. Käytettävyystutkimuksella hankittiin kvantitatiivista ja kvalitatiivista tietoa tuotteesta.</p> <p>Laite jolla ohjelmistoa käytetään vaikuttaa ohjelmasta saatuun käyttökokemukseen merkittävästi. Hyvän käyttökokemuksen saavuttamiseksi lopullisen järjestelmän käytön tulisi olla sujuvaa. Aikaisten prototyyppien kokeilu ja kohdelaitteiden saatavuus ovat tärkeitä tekijöitä siirtymäprosessin kannalta. Yleisiä ongelmatilanteita ja haasteita joita kohdattiin suunnittelu- ja toteutusvaiheissa on listattu työssä. Loppukäyttäjän kannalta useat ongelmat olivat rasittavia ja vaikuttaisivat käyttökokemukseen negatiivisesti jos niitä ei korjata. Uuden käyttöympäristön tuomat ongelmat joudutaan ratkaisemaan aina uudestaan. Vain kokemuksella vastaavista tilanteista on merkittävästi etua itse ratkaisujen löytämiselle.</p> <p>Toteutetun kosketuskäyttöliittymän tuen voidaan todeta vastaavan sille asetettuja tavoitteita ja vaatimuksia hyvin; se mahdollistaa ohjelman käyttämisen kosketuskäyttöliittymän omaavissa laitteissa ja kaikkein merkittävimmät käyttöliittymäelementit on tuettuina.</p>			
Avainsanat: HCI, BIM, kosketus, migraatio, tablet, sormitietokone, katseluohjelmisto, arviointi			

Acknowledgements

I found myself thinking about a thesis subject that would enable me to write about something I am both interested and knowledgeable about already. I want to thank Tekla, and especially Ville Rousu, for the opportunity to write this book and thus finalise my studies.

I want to thank my supervisor Lauri Savioja for the constructive criticism and good feedback he was able to give during writing. I thank my instructor Marko Myllymaa for continued support around the clock. I want to thank my parents for their endurance for encouraging me to finish my diploma, and many thanks to my friends who were there supporting me and reflecting my thoughts and visions with me.

Table of Contents

Acknowledgements i

Abbreviations vi

List of Figures vii

List of Tables viii

1 Introduction 1

 1.1 Research Objectives 1

 1.2 Structure of the thesis 3

2 Background 4

 2.1 Input mechanism properties 4

 2.1.1 Mouse 4

 2.1.2 Touchscreen 5

 2.1.3 Stylus 5

 2.1.4 Trackpad and Touchpad 6

 2.1.5 Comparison 7

 2.2 Touch interaction environments 8

 2.3 Touch input technologies 9

 2.3.1 Resistive touch panel 9

 2.3.2 Capacitive touch panel 10

 2.3.3 Infrared LED array 10

 2.3.4 Optical imaging with video camera 11

 2.3.5 Optical imaging with infrared sensors 11

 2.3.6 Hardware manufacturers 12

 2.4 Support in operating systems 14

 2.4.1 Microsoft touch support 14

 2.4.2 Apple touch support 15

 2.4.3 Android 16

 2.4.4 Linux multi-touch support 16

 2.4.5 Operating systems in summary 17

2.5	Common problems analysis	17
2.6	Usability	21
2.6.1	Touch interface usability metrics	21
2.6.3	Cognitive and physical load	22
2.6.4	Perceived performance	23
2.7	Validation methods.....	24
2.7.1	System Usability Scale	24
2.7.2	Affinity diagramming.....	25
2.8	Building Information Modeling	26
3	Defining requirements.....	28
3.1	Functional requirements	28
3.2	User stories	29
3.3	Targeted hardware.....	30
3.4	Main features	31
3.5	Technical background of Tekla BIMsight	33
3.6	Technical limitations	34
3.7	Requirements for the UI.....	37
4	Implementation	40
4.1	Iterations of touch support.....	40
4.1.1	1st prototype: Joystick buttons.....	42
4.1.2	2nd prototype: Touch event translation to mouse events ...	44
4.1.3	3rd prototype: Full multi-touch implementation	48
4.1.4	4th prototype: Using only WPF raw touch events	51
4.1.5	5th prototype: Performance optimization iteration.....	54
4.2	Delivered functionality of the Tekla BIMsight 1.4	57
5	Usability evaluation	59
5.1	Planning the evaluation	59
5.1.1	Evaluation goals and participants	61
5.1.2	Limitations	62

5.2	Execution of the evaluation	62
5.2.1	Inspection briefing.....	62
5.2.2	Inspection visit.....	63
5.2.3	Feedback for the participants	65
5.2.4	Debriefing the user testing	65
5.3	Analysis of the material	65
5.4	Findings from the evaluation	67
5.4.1	Touch interface was easy to pick up and accepted	67
5.4.2	Mental model of navigation	67
5.4.3	Orbiting tool lead users into error situations easily.....	68
5.4.4	View angle is very narrow inside buildings	68
5.4.5	Difficulty of discovering the Tap&Hold gesture.....	69
5.4.6	Accessing part details was difficult.....	69
5.4.7	Measurement tool did not match mental model.....	70
5.4.8	Reception of the zooming controls	71
5.4.9	Other observations.....	72
5.5	Findings from the affinity diagramming.....	73
5.6	Findings from system usability scale	74
5.7	Discussion on the usability test	75
6	Results.....	77
6.1	Summary of prototyping	77
6.2	Lessons learned	79
6.2.1	Prototyping of unknown technology	79
6.2.2	Indirect interaction is difficult with direct input	79
6.2.3	Direct input — experience is critical.....	80
6.2.4	Tricks in handling the implementation of touch.....	81
6.2.5	Rendering performance.....	82
6.2.6	Unknown safe exits in navigation.....	83

7 Conclusions84

8 Future Steps87

References88

Appendix A96

Appendix B97

Appendix C98

Appendix D.....99

Abbreviations

AEC	Architecture, Engineering and Construction business
APDT	Agile Planner on Digital Tabletop
API	application programming interface
BIM	Building Information Modeling
BSI	buildingSMART International
CAD	computer-aided design
DLU	dialog box unit, a measure used in Microsoft Windows
HCI	human-computer interaction
IAI	International Alliance for Interoperability
IDE	integrated development environment
IFC	Industry Foundation Classes
ISO	International Standardization Organization
LCD	liquid crystal display
LED	light emitting diode
MSAA	Microsoft Active Accessibility
MSDN	Microsoft Developer Network
PC	personal computer
RPC	remote procedure call
SDK	software development kit
SUS	System Usability Scale
UI	user interface
WPF	Windows Presentation Foundation
XAML	extensible application markup language
XML	extensible markup language
ZCR	Zone of Comfortable Reach

List of Figures

2.1	Touch-enabled operating systems on a timeline.	17
2.2	Illustration of the changes in finger pose when using a wall display (left) and when using a tabletop device (right).	22
2.3	Responses in Windows 7 for single..	23
2.4	Responses in Windows 7 for double tapping.	23
2.5	Tekla BIM workflow.	27
3.1	Colouring of entire models in Tekla BIMsight.....	33
3.2	Colouring of entire selected parts in Tekla BIMsight.	33
3.3	Screen estate available to the 3D view in different devices.	37
4.1	Areas in the application UI that require attention for touch support.....	40
4.2	Graphical comparison of the turning behaviours.....	46
4.3	Illustration of different gestures and how the dot product distinguishes them.....	53
4.4	Normal mode for the user interface.	58
4.5	Tablet modes for the user interface.....	58
5.1	Visiting the construction site.....	64
5.2	Team creating the affinity diagram.	66
5.3	Press and wait indicators in Microsoft Windows 7.....	69
5.4	Press and wait indicators in Microsoft Windows 7.....	69
5.5	Press and wait indicator in Tekla BIMsight.....	69
5.6	User attempted to use the zooming joystick as the visual clue suggests without achieving perceptible results.....	71
5.7	Detailed drill down of the SUS results.	74

List of Tables

2.1 Input device comparison..... 7

3.1 Navigation controls32

3.2 Touch input emulated as mouse input.....36

4.1 Test durations and numbers of calls to respective update
methods.56

1. Introduction

Designing a user interface (UI) for touch based input interface requires more attention than pointer based input systems, such as the (computer) mouse [1, 2]. For nearly three decades the mouse and the keyboard have been the dominant input methods for computer assisted design. Touch-based user interfaces are promised to bring a more natural user experience and portability that cannot be rivalled by previous input devices. Now touch screens are becoming more and more popular and software vendors are looking to take advantage of them.

The tablet computers are blooming in the computer industry [3, 4] and they have been used in the field in heavy industry for some time already for inspections, taking notes and following up the site. Construction industry is just now beginning to harness the potential of tablets and there are not many applications that support this usage scenario well. What capabilities do the tablet computers have over conventional laptops that make them so compelling? The direct input ability and absolute positioning of fingers enable more direct design paradigms to be applied on the application design, which has been argued to add gains in efficiency [1]. Ergonomics of a device designed for touch or styli are different from that of a mouse.

1.1 Research Objectives

The project for this thesis is to unite the worlds of small touch screen or stylus devices with desktop environment in the same software. The software is intended to support styli and touch on tablet computers. Meanwhile the user experience on desktop computers should be maintained at the same level or improved. This thesis will try to address the general problem of migrating desktop software into a touch input environment.

There are three research questions defined for this thesis. They are as follows:

1. What quantitative measures may be applied to specialized software in order to measure its productivity and usability?
2. How should software be designed so that multiple input mechanisms may be supported?
3. What is efficient use of touch-based user interface and are there means of measuring it?

The project of implementing the touch input into Tekla BIMsight [5] represents the practical case study of this thesis. That is about studying the implementation of the support, evaluating the results from usability point of view, and to document future steps from that point onward. The main research objectives are:

- Identify and document the problems that impede Tekla BIMsight from being used on touch input devices.
- Study and document the implementation work to support touch input devices.
- Document the implemented support.
- Validate and analyse the implemented level of support. Analysis will concentrate on the efficient use and productivity.

The main outcome of this thesis is a documented solution for a migration of desktop application to a tablet device that is intended to be used outside of office environment. The solution will be built onto Tekla BIMsight as support for tablet usage with stylus or fingers. Prototype solutions need to be built and refined to form the final solution that will be published as part of the next application update.

The actual support for touch input will be a team effort made by the team responsible for the implementation of the application. Most of the practical work committed for the completion of this work is both contribution to the team's efforts and this thesis. The documentation concerning this thesis and the scientific background research are solely part of the thesis work.

1.2 . Structure of the thesis

First the thesis will explore the general problem domain and shed light on the usage of the application in the background section. That section also contains the general problems that other related works point out. After that the problem domain is localised into the Tekla BIMsight application. The problems faced during the implementation are studied. The effects of the implementation are validated and tested with real users and the results from that study are documented. Finally a conclusion is drawn from the project, its success and future steps for continued study. Results are presented in the final chapter with reflection.

2. Background

2.1 Input mechanism properties

Understanding different input mechanisms and their properties will enable us to study the common problems that they pose. Different kinesthetic and ergonomic properties that are involved will have a significant effect on the overall *'feeling'* of the user experience. Physical properties of different input methods are evaluated so that the question of relevant support would be clearer later on.

2.1.1. Mouse

Mouse as an input mechanism is an indirect pointing device. When user moves the device a virtual pointer moves on the screen according to the programmed logic. The cursor speed is nowadays determined with ballistic algorithms [6], so that it is easier for user to hit even 1x1 pixel targets with gentle movements and also rapidly move across larger distances. The indirect nature of mouse is present also in the fact that the mouse cursor may be moved without taking action on the user interface. Moving the mouse cursor over a control is called hovering and usually the system displays some in-place help, or tool-tip help, for the user when hovering over controls.

The mouse is also a non-haptic control as the common mouse does not inhibit any feedback mechanisms. A force feedback mouse device was developed by Salcudean in 1993 [7], but they did not gain much popularity as, for example, the wheel mouse did. The generic mouse nowadays has two physical buttons, left and right, and the pressible scroll-wheel as the middle button. A modern mouse using PS/2 connection reports movement at 10-200 times per second [8] and gaming mice connected to universal serial bus (USB) at a staggering 1000 Hz intervals with accuracy of even 5700 dpi [9].

2.1.2. Touchscreen

Touch input over a display device is a direct input mechanism where input is active when fingers touch the surface of the screen. Input is targeted at the absolute coordinates of the finger relative to the image on the screen. When a finger is pressed onto the screen it hides the content directly underneath it. This is called occlusion and it presents the main disadvantage of touch screens. Touch input mechanisms do not support hovering over controls. The directness lends to the fact that any input, even unintended, is still valid input. The directness is also referred to as being more natural [2] for the user: *“Touch provides a natural, real-world feel to interaction. Direct manipulation and animation complete this impression, by giving objects a realistic, dynamic motion and feedback.”* [2] It is more haptic than the mouse generally is but commercially available touch screens do not yet have simulated textures, so the haptic sense of fingers is not fully utilized. There is however ongoing research for this field as well [11]. The whole device may be vibrated when a successful input has been registered to provide feedback, as it is done in some mobile devices [10].

2.1.3. Stylus

Stylus input as implemented by Wacom or N-Trig utilize an active stylus for input mechanism. Styli come integrated into the display device or with a separate recognition panel for the stylus. Some devices are also able to detect touch and stylus [12, 13, 14, 15]. The device is able to recognize the location of the styli and the pressure of the contact point with a high resolution [12]. The ability to use a pen to draw on a computer screen has made the styli very popular among artists who may use it as a regular pen. The behaviour of the styli may differ according to manufacturer, but usually the styli support hovering to move the mouse cursor on the operating system and to register input when the tip is pressed down on the screen.

Styli may be configured to behave with a relative movement mapping or an absolute movement mapping. When configured to use the relative movement mapping the stylus resembles the mouse very

closely as the cursor is dragged with the stylus and pressed when the styli hits the panel. If the stylus is mapped with an absolute movement mapping then using stylus behaves almost as touch but with a better input resolution because of the smaller contact surface.

Most styli feature a function button on the side and the other end of the stylus may be used as an eraser tip. Depending on the driver support the functionalities may be altered to suit the user's needs.

2.1.4. Trackpad and Touchpad

A trackpad features a panel that is able to detect finger touches. This device is also known as a touchpad. When user just taps a finger onto the panel it is recognised as a click event. When a finger is dragged across the panel it is interpreted as cursor movement. Trackpads often come with physical buttons for mouse functions close to the trackpad, situated in front of the trackpad. These devices have been limited in bus bandwidth in PS/2 interface that has been in use for mouse devices for some time. Trackpads using this interface have limitation of two fingers to be real-time tracked, and limitation of four in recognized finger contacts. [16, 17]

Modern trackpads feature recognition of multiple fingers and operating systems have begun to take advantage of this. Other gestures on the trackpad are for example scrolling, which in some cases may be invoked with two fingers dragging on the trackpad, dragging a single finger on one edge of the trackpad or dragging a finger in a circular motion on the trackpad. Synaptics is one manufacturer of these devices and they report that the latest models feature 1000 dpi accuracy when using their improved internal bus to connect the device to the system. A more modern approach of the trackpad is the external mouse device called Apple Magic Trackpad [18], which has multi-touch gesture support in addition of being a very big trackpad.

2.1.5. Comparison

Of the devices presented above the mouse and the trackpad are the only indirect input devices. A stylus may be both a direct and an indirect input device depending on the solution as the system may reside directly over a display device or just be an external device for a desktop computer making it indirect. Trackpads are always indirect as the touch sensitive area is always a separate device from the display device. They represent a go-between solution between the stylus and touchscreen in the sense that fingers may be used without the need for a specialized pointing device, but gestures and the directness of the touchscreen are not utilized. It is more like a mouse than touchscreens are. Table 2.1 lists different devices and their main categorical features.

Table 2.1: Input device comparison.

Device	Direct	Indirect	Mistake Likelihood ↓	Accuracy *	Update Frequency
Mouse		X	Rare	1000-5700 dpi	100-1000 Hz
Trackpad		X	Happens	1000 dpi	100 Hz
Stylus	X	(X)	Less common	2540 lpi	133 Hz
Touchscreen	X		Common	1000-4000 dpi	Depends

* Accuracy for stylus is in lines per inch and other devices are in dots per inch.

Update frequencies on each device are high in most cases. Considering those devices in wired connection with the computer, the update frequency might depend on the connection. Wired devices or integrated hardware usually bear the possibility of utilizing as much bandwidth as is available. Touchscreen update frequency might pose another kind of problem if the amount of fingers reported affects the update frequency directly and that is why Table 2.1 indicates this speed to be varying. Ambiguous input will be discussed later in section 2.5 where studies [1, 19] indicate that mistake likelihood is higher for direct input devices than indirect input devices. Mistake likelihood is an estimate based on the studies where different mechanisms were compared.

2.2 Touch interaction environments

Tablet computers are handheld computers that are operated with touch or stylus input. The device is roughly similar to any laptop available on the market, but it is missing the typical input mechanisms like the keyboard and the trackpad or IBM's track joystick. Tablet computers ship with special drivers or software to accommodate for the lack of a physical keyboard and offer — in similar fashion to smartphones — a virtual keyboard for text input. Touch UI is the latest trend in smartphones and most of the new phones reaching the market are touch interactive instead of physical keyboard. This has brought touch technologies to the consumer markets.

Tablet/Laptop convertibles are portable computers with a swivel joint between the screen and hull of the computer. The screen may be folded display downwards for transportation and opened to operate in a laptop mode. The screen may also be folded on the hull with the display facing outwards transforming it into a thick tablet computer. One such example device is the Dell XT2 [15]. Other tablet/laptop computers like the Asus Transformer Prime [20] use the keyboard dock to transform the tablet into a laptop computer.

Tabletop is used to describe a touch interactive computer in which the image is projected onto a tabletop — hence the name. Tabletops may combine different technologies in order to achieve the result but a common approach is to project the image using one technology and detect touch using another. In section 2.3.5 a novel approach to tabletops is introduced. Similar devices may be mounted on walls to turn them into interactive wall displays. An interactive wall may as well be achieved by projecting or rear-projecting an image onto the wall and detecting when users fingers touch the wall to enable interaction. Tabletops and interactive walls range in size from single user devices utilizing the increased screen estate to multi-user setups taking advantage of the collaboration aspects. In tabletop multi-user environments, the design of the content displayed on the screen should reflect the intricacies of the environment.

Wang, Ghanam and Mauer migrated Agile Planner [21, 22] software to be used on SMART Technologies digital tabletop and touchwall [23]. They note in their experience, that there were few studies available on migrating desktop software to digital tabletops. In other works their own work has been mentioned a few times as reference on using Windows Presentation Foundation for their implementation of Agile Planner on Digital Tabletop (APDT) [21]. This work is especially interesting because the platform is the same as in this project.

Their interest in the project was on multimodal input system (i.e. touch, gestures, handwriting and voice). APDT was intended to change traditional practices on tabletop planning meetings. They noted that the advantages of computerized environment were good. They anticipated in the beginning of the project that their lack of handling several touch points at the same time might have repercussions later in the project, and found it to have severe influence on the overall perceived performance saying *“People actually could notice that this limitation was hindering the interaction that would usually be more dynamic in traditional mode.”* [21] This seems to point out the human ability to give great value for small details. Lastly they found that people preferred to use keyboards instead of touch for text input saying *“using a keyboard is regarded more “natural” in a computer environment”* [21], which might indicate that the tactile feedback is appreciated.

2.3 Touch input technologies

2.3.1. Resistive touch panel

A resistive touch panel is made of two layers that have electrically resistive coating and which are separated by a thin space. The layers are coated so that the coating faces the other layer and that the coating forms lines horizontally on the other and vertically on the second. The control logic is able to sense the variation in the resistance between the two layers when an object, such as finger, is pressed against the panel. The horizontally aligned layer represents the vertical position and vice versa. The benefits of resistive technology include low cost of

manufacturing, substantial resistance to the elements, and usability with pens and other pointed items. The downside is that the panel may easily be damaged by sharp objects. Resistive technology is limited in the number of points detectable on the panel and most of the devices only support one touch point. This technology is still often used in restaurants and places where liquids may be spilled onto the surfaces.

2.3.2. Capacitive touch panel

The capacitive sensor grid operates on a similar principle as the resistive grid, in that it has an electrically conducting coating on it, but uses the conducting quality of human skin to sense the distortion in the electrostatic field of the panel that is measurable in capacitance. The benefit in this technology is that it can sense a more subtle touch than resistive panel would sense, because the electrostatic field of the panel is altered when the finger just barely touches the panel. From usability point of view this is very good. The downside is that the panel cannot be used with any objects that do not conduct electric current, and this is especially present in cold weather, when attempting to use the panel with gloves. There are different variations of this technology such as surface capacitance, projected capacitance, mutual capacitance and self-capacitance [24]. Capacitive technologies support multiple touch points, with different limitations in different variations of the technology. This seems to be the main technology present in modern smartphones that have touch-screen panels.

2.3.3. Infrared LED array

Infrared LED (light emitting diode) grid is a technology where arrays of infrared LEDs and photodetectors are aligned to the edges of the display device to form a sensor net over the display [25]. Two such devices are the HP TouchSmart integrated computer [26] and Dell SX2210T Flat Panel [27]. Dell refers to this as optical touch technology [27]. When any object is placed onto the net the sensors detect that some light sources are no longer visible. This can be translated into a location and possibly the size of the object pressed against the display. The major disadvantage with this technology is that it cannot

distinguish between several touch points reliably, because the points may too easily occlude each other on the grid thus resulting in false detections.

2.3.4. Optical imaging with video camera

Infrared light may also be used in optical detection and there are a few variations to this. The first one uses a construction where the picture is projected onto an opaque screen either from behind the screen or from above. Under the screen there is an infrared light source, such as a bulb or LEDs, which emit infrared light through the screen. When the light hits any objects that reflect it back, the reflected light is sensed by the video camera that is placed under the screen. The pictures seen by the camera are sent to the control logic which uses imaging technologies to sense where the fingers, palms and arms of the user are and reports these to the software. This approach has virtually no limitation in the number of touch points it can detect and it is able to detect other things reflecting back from the screen such as tags and shapes placed on the screen. The limitations of this specific solution is that it requires space for the construction as there has to be enough distance for the camera to see everything that is happening on the screen. Touchlib is a good example of an open source software platform that enables building of such a system [28]. Such systems are commercially available from, for example, Microsoft Surface [29] and Multitouch.fi [30].

2.3.5. Optical imaging with infrared sensors

A later innovation called PixelSense [31], developed by Microsoft Research, Applied Sciences Group, has a similar approach to the optical sensor presented above, but without the optics of the camera or the space requirements. It has a conventional liquid crystal display (LCD) element that forms the image displayed on the screen. Underneath that is the backlight that has pairs of white LEDs and infrared LEDs. Above the LCD element is the sensor layer that has photoelectric sensors for detecting light reflecting back onto the screen. The control logic collects the individual values and forms a

picture that can then be evaluated using image detection algorithms as with previous technology. This technology has the benefit of producing large displays that have virtually no limitations for the number of detectable touch points and it makes this ideal for multi-user products. One disadvantage is that the imaging technology requires quite a lot of processing power. Another disadvantage is that this technology is very new and immature and the cost of producing the display unit is still rather high, and there is only one producer at the moment. This technology is interesting but irrelevant at the moment when considering tablet products since it is not applicable there yet.

In a way a similar approach is also used in the Microsoft Touch Mouse that utilizes a capacitive sensor grid for detecting the touch but then transforms this in the driver into an image that is then processed with image recognition algorithms. This is in addition to the indirect movement produced by moving the mouse on a surface.

2.3.6. Hardware manufacturers

There are several hardware manufacturers for touch panels and touch input systems such as Wacom, N-Trig, Samsung, 3M, Multitouch.fi and Synaptics to name a few.

Apple was the first to include multitouch support in their laptop line and after that other manufacturers, such as Synaptics have been providing similar solutions with their hardware and drivers as well. Apple as a hardware manufacturer is known for its famous touch devices, such as the iPhone smartphone and iPod MP3 players. The iPhone was debuted in 2007 and it was noted as the most successful smartphone on the market.

Samsung is a South-Korea based manufacturer of several touch-enabled devices that range from cell phones and tablets to televisions and other touch devices. Lately they partnered with Microsoft to produce the new technology called PixelSense [31] and are now the sole manufacturer of the Microsoft Surface 2.0. It is able to distinguish 50 individual points simultaneously. They are in a significant role on

the touch device market and they employ several different technologies in their products.

Synaptics is the hardware manufacturer that has been supplying probably the most touchpad devices to laptops around the world. Their products mostly utilize capacitive technology for sensing finger movement on the touch device and so far multitouch gestures have not been widely available in Windows products. [17]

Wacom was the first in the market with stylus devices and only lately taken part in the touch-enabled market. Their main target segment still remains the artistically talented market whose users want to have powerful tools to use with their styli. Wacom Bamboo products are able to distinguish 16 distinct touch points [32].

N-Trig devices are sub modules for LCD-panels and they provide driver support for Windows, Linux and Android operating systems. They were established in 1999 in Israel. Their product is used in, for example, the convertible tablet-laptop Dell XT2 [15] and Fujitsu STYLISTIC Q550 [14].

Computer vendors usually license their products to use one manufacturer and they provide the vendor with the proper operating system drivers to integrate to their computers properly. Such vendors are Dell, Acer and HP among others.

SMART Technologies is a hardware vendor specialising in touch input driven whiteboards and panels that integrate with plasma or LED displays [23]. Their products utilize Optical imaging with video cameras at the sides of the whiteboards.

2.4 Support in operating systems

Different technologies presented above are applied in various environments by different software vendors and hardware manufacturers. Below are listed some solutions that are commercially or freely available on the market at the moment.

2.4.1. Microsoft touch support

Microsoft often showcases their tabletop computer projects Surface and Surface 2.0, when referring to multi-touch application development. The Surface project was a research project primarily although it is now available commercially as Samsung SUR40 [33] for Microsoft Surface. It runs Embedded Windows 7 internally but the software solutions are implemented against the software development kit (SDK), Surface SDK 2.0 [34]. The Embedded Windows 7 has two operation modes compared to the regular installation; Surface mode where applications run in fullscreen and all notification windows are suppressed, and Windows mode where they have full access to Windows functionality. This enables execution of all normal software on the Surface as well as the touchable [2] software especially designed for the environment. Since the operating system on the Surface is Windows 7, all other touchable [2] software that runs on Windows 7 runs on the Surface too.

Microsoft Corporation is one of the commercial vendors to bring touch device support to a business market, where others may build and develop on their platform with off-the-shelf products. Microsoft produces hardware and also forms partnerships with hardware vendors to produce products that benefit their operating system. Touch support is possible in Windows Vista [2] to some extent, but Windows 7 was their first operating system to offer full multi-touch support [2].

Microsoft offers Visual Studio™ product line for developers as the supported development environment. Visual Studio built-in language support enables programming in C/C++, VB.NET (Visual Basic .NET), Visual C# and Visual F# by default. Other languages may later be

added as extensions for Visual Studio. The suite of tools includes integrated development environment (IDE), compiler, debugger, extensions support, authoring tools for creating installers and other tools. Microsoft offers Visual Studio in several bundles depending on the suite that the developer requires. The “Express” line of Visual Studio products is free development environment, without the extensions support, that Microsoft distributes through the DreamSpark service. Visual Studio Professional is the standard commercial version offered for professional developers. Ultimate and Team System versions add tools for metrics and developer communities. [35]

2.4.2. Apple touch support

Apple Corporation has become well known touch device manufacturer ever since they debuted with their first iPhone in 2007. What is at times lost is that they have significant experience from building touch-enabled devices from 2002 already, with the first iPod that featured a touch-sensitive wheel controller instead of a scrolling wheel.

In 2008 Apple and their operating system OS X boasted a multi-touch trackpad on their MacBook Air notebook. The trackpad was new because it brought the multi-touch input to computer users with an off-the-shelf product and enabled them to use their fingers to pinch, swipe, rotate and zoom to access features on the applications. These were however limited to certain functions and were not made available system wide yet. [36, 37]

Developing software for Apple iOS™ or Mac OS X™ systems requires the Apple Development suite called Xcode that contains all the tools needed to develop on the platform. The suite includes an integrated development environment that enables easy development and debugging of new software. Tools included are IDE, compiler, debugger, authoring tools for installations, publishing tools and other tools. The supported programming languages in the Xcode environment are C, C++ and Objective-C. The development tools are available for a yearly fee on the developer program. [38, 39]

2.4.3. Android

The first Google Android device to hit the market was in October 2008 when the T-Mobile G1 was released in the United Kingdom [40]. The device did not offer much when it was released in terms of touch-enabled control mechanisms, because the multi-touch features had been disabled on the first release, supposedly to avert patent infringements with Apple.

Android platform features a rich and open-source SDK that supports developing mainly in Java programming language, but in special cases native libraries may be created in C++. Android devices range from smartphones to tablet computers with various hardware configurations.

2.4.4. Linux multi-touch support

Linux is known as an operating system that comes in various distributions. Linux is developed as an open source initiative and this makes the distinction between any Linux based operating system a little difficult. Any support that Linux then has, can be attributed as either a generic support that can be integrated into any distribution, or something that comes as out of the box functionality with a specific distribution.

Canonical is a company that specialises in the development and support of the Ubuntu distribution of Linux, especially they target corporations in selling the enterprise solutions of Ubuntu that range from cloud computing clusters to secure laptops running Ubuntu. Now they are engaged in the multi-touch support development for Ubuntu. In October 2010 with the release of Ubuntu 10.10 codenamed the Maverick Meerkat, they also announced the uTouch library, which featured a full gesture library for recognizing gestures performed by the user [41]. This enabled the users to quickly move, maximize and restore their windows or call up the dash by performing multi-fingered gestures over a multitouch display.

Later on with the release of Ubuntu 11.04 the full support for multitouch input was added, even for legacy applications that did not support it. This support was achieved with a mid-ware service that sits close to the windowing system and communicates actions to the applications. It is called Ginn and it enables mapping touch gestures to specific features in the applications with a simple mapping configuration in an extensible markup language (XML) file. Ginn is short for "*Gesture Injector: No-GEIS, No-Toolkits*" [42, 43].

2.4.5. Operating systems in summary

Figure 2.1 below illustrates when different parties enter the world-wide market of touch or multi-touch-enabled devices. In the figure the start-ups related to Apple and Google are highlighted. Microsoft Research is the institution where all of Microsoft's research is done and thus it is difficult to know how long they had been working on touch applications. It is also interesting to compare the time periods that are associated with the experience and development on different time periods before a major release. Equally interesting is to see when the first major releases are in relation to other timelines. Linux Multipointer X is contradictory in the sense, that it required a lot of knowhow from the user to take the release into use.

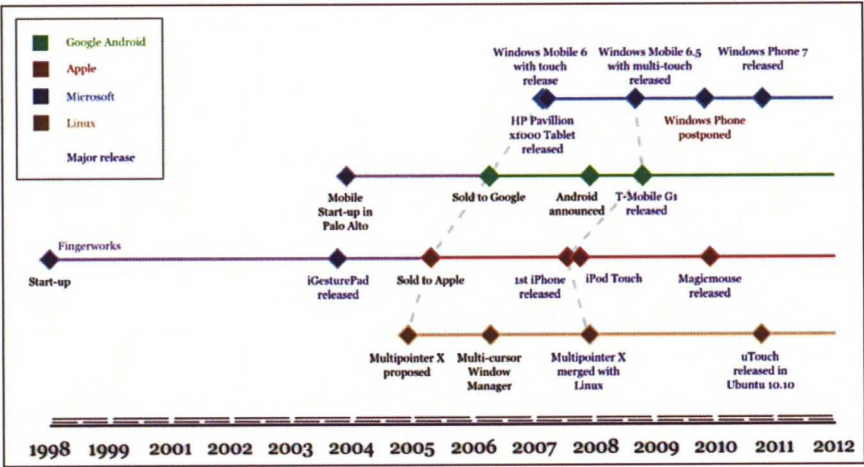


Figure 2.1: Touch-enabled operating systems on a timeline. [44, 45]

2.5 Common problems analysis

Several experimental studies have been conducted on touch input systems that choose usually several research questions to test in their experimental setting. These studies contain well quantified research material that may be applied to interface design later on. However, the studies lack a good listing of common problems or a quick summary of them. Ryall et al [19] showed good initiative when they listed out in their experience the most of the relevant problems concerning tabletop computers — I think those problems also concern tablet computers too, although maybe not in completely the same extent. Since then a few more papers [1, 46, 47, 48, 49, 50] contribute valuable insight to the problems, but do not bring much more problems on the table.

Following is a listing of the problems pointed out by many authors [1, 19, 46, 47, 48, 49, 50, 51], that will be evaluated in relevance and seriousness to the case of Tekla BIMsight. This is done in order to understand their true meaning concerning this context and to refine the questions that need answering when designing the software in question later on.

Occlusion [19] affects all touch based displays and happens whenever user attempts to touch the screen. Some solutions circumvent this feature by project a picture onto the user's hand and utilizing that area [46]. On devices where the displayed picture is on the device and not projected, occlusion is inevitable.

Reach of the user or the Zone of Comfortable Reach (ZCR) as explained in the paper about Magic Desk [47], is how far the user affects mostly tabletops. The question of reaching items may rise with tablets as well if the grip of the device is very important and the reach of user's fingers, especially thumbs, is important while retaining the grip on the device.

Forlines et al question if the advantages of proprioception, the ability to track body parts kinesthetically, are counteracted with the problems of occlusion and reach [47], saying: *"From a human physiological*

standpoint, proprioception, one's inherent ability to keep track of the location of one's body parts kinesthetically might be expected to result in significant advantages for direct-touch bimanual input; however, it is unclear whether occlusion and the reaching over large distances on a tabletop will counteract this benefit."

Clutching [49] is the activity of returning the pointer device to earlier position in order to move the virtual cursor further on the display device. This is apparent on relative trackpads on laptops where user needs to move the cursor with a low control-to-display ratio across the display. Same problem is also present with a control interface of a 3D model which is trajectory based movement. To rotate the model around, user needs to clutch several touch gestures to make up a more significant total movement. There are studies for reducing the need of clutching [49, 51].

Tools are always made to solve certain problems the users face. Designers should be able to decide what paradigms they want use in the solutions, when they design these tools for the user interface [19]. Accot et al compared the differences in quantitative performance between trajectory based applications to point based selection performance [50]. It is a good reference as to how the design principle of a tool may be affected on the chosen paradigm that it implements.

The second big problem related to the paradigms mentioned above, is the order of design and implementation for the different control mechanisms. A briefing by Microsoft for developers that are starting out with touch interface design, [2] suggests, that developers go about the problem on a touch first basis. Mouse and pen devices have more limited ways of expressing gestures when compared to touch based systems. For example, a generic wheel mouse has three buttons and the mouse wheel, and styluses generally have one or two distinguishable tips and one or more buttons for alternate behaviour. Gesture recognition systems may be built for both devices, but still multi-touch input has more expressional power with a single finger. Even more complexity is added to the touch system design when

flicking, dragging, pinching and rotating gestures are considered, raising the complexity by a factor of three at least.

Finger resolution [19] may mean many things. On a low level, it could be interpreted to mean the resolution of identifying single touch points. On a bit higher level of analyzing multi-touches we could interpret it to mean the amount of fingers that the device is able to distinguish or maybe even the pose of those fingers that are touching the device surface.

Ambiguous input or unintended input [19] is the principal problem of direct touch systems. Users must understand that all touching of the system will affect the system if the system is built to react to all touches. However, the designers of the system should also take into account what it is that the system does by default with the touch. Direct touch control is in essence a modal control mechanism and a paradigm shift from the indirect control of a mouse controlled system.

The problems presented above concern in some or all cases tablets as well as tabletop systems.

Tabletop computers inhibit some very distinctive problems that are partly problems on tablets as well. To name a few more issues specific to tabletop; multiple users are hardly a problem for a tablet computer, which may be easily passed to the other user for review, so **multi-user coordination** [19, 21, 22, 48] is more related to tabletop systems. **Orientation** [19] is also the problem that persists more on tabletop systems with multiple users, rather than on tablet computers. Unless the application is especially built to utilize the orientation, which nowadays is, more often than not the case like, for example, with panoramic photographing applications.

Crowding and cluttering [19] are tabletop problems, but might affect software that encourages users to approach a touch-enabled wall projection.

Ergonomic issues [19] affect all devices and all form factors. Designers and developers should consider the ergonomics of control

placement and refer to guidelines set for different devices by the manufacturer in their developer documentation. Tabletops also suffer from the ergonomics of the furniture as noted in their study of tabletops [19, 46, 47, 48].

Text input [19, 21] is deemed difficult on touch based devices for several reasons. People sense with their hands far more than they do with their eyes with the device they use for typing text. There even is a device for the Apple iPad that offers users the tactile feedback of a keyboard on the virtual keyboard of the iPad [52]. Other studies pointed out that users preferred a wireless keyboard in a computer environment over the handwriting recognition system of the APDT [21]. The project group on APDT suggest even to have wireless keyboards available for the users for faster text input [21].

2.6 Usability

2.6.1. Touch interface usability metrics

There are interesting studies made on the Human Computer Interaction (HCI) with input methods [1, 19, 21, 46, 47, 48, 49, 50, 51]. HCI has been studied to understand which input method is the best at different tasks. Forlines et al studied the performance between mouse and direct-touch on quantitative performance and subjective preference. Their experiments were designed to test the two devices for unimanual and bimanual scenarios. They point out the key problems in user interface design for touch devices that are furthermore supported by Fitts law, that hitting a target that is smaller than the device targeting it is very difficult [1, 53]. They acknowledge that mouse may be better at some tasks than direct touch and raise the question of *“the appropriateness of a direct-touch tabletop interface for a single user working on tasks requiring only single-point interaction”*. They point out that there are physical challenges on tabletop displays when dealing with recognising finger touches, because the touch area, on the device changes when user reaches for items that are further away from the user as illustrated in Figure 2.2.

The study does not coincide with this project's premises, but the problems faced in that study are relevant.

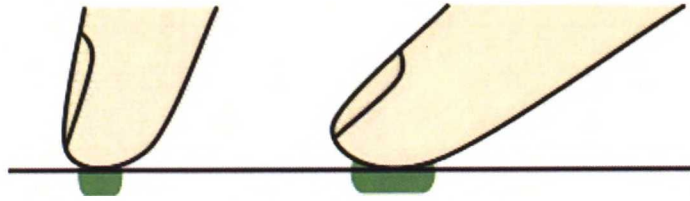


Figure 2.2: Illustration of the changes in finger pose when using a wall display (left) and when using a tabletop device (right).

Magic desk [47] developers note in their work the absence of migration studies as well. They found little studies investigating the integration of multi-touch in desktop environment and set out to investigate the integration of multi-touch controls on users' desktop and the optimal positioning of the control surfaces. They found, that the multi-touch displays commercially available on market today are actually the worst configuration for the users: *"The vertical screen is a poor region for performing one-handed tasks."* They base their experiments on ergonomic design and on study of digital tabletop usage.

2.6.2. Cognitive and physical load

In software production context efficient use is often described in the sense of usability or user experience. It may also be defined in the sense of industrialization, when it is measured on the performance of an individual and how many work units they produce. It should be no surprise then, that both worlds share a common ground in the theory.

Physical load is that of using the UI and having to cope with the inaccuracy of the control device used i.e. the control size versus the finger accuracy that comes from users' finger size.

Cognitive load is the load user has to bear when they have to remember where items are displayed on the UI, where some of the control mechanisms may be hidden for reducing the physical load on the UI.

These two measures are tightly bound together, so that when another one is decreased by some changes in the UI, the other usually

increases. A very low measure in physical load usually translates to a larger cognitive load and vice versa.

2.6.3. Perceived performance

Perceived performance is the experience user gains from the product that is not necessarily related to any otherwise measurable metric. It is the qualitative vision of the user's expectations and how the product fulfils them. It is very difficult to show that the perceived performance is good, but it is possible to show by ways of user testing that perceived performance is not at an acceptable level.

Nielsen's usability heuristics [54, 55, 56] contain several similar elements that deal with accounting for perceived performance. That is, how can you design software so that the user will have a good impression after using the software?

One measurement that may be used for perceived performance is the responsiveness of a given UI. If a UI is always responsive to the user's interactions it is perceived to be so, and it contributes to the overall good impression of the performance of the UI. Responsiveness may in this case also be the immediate response of anything, such as giving a rolling circle that lets the user know of a background processing going on. Microsoft Windows 7 single tapping and double tapping gesture responses are illustrated in the Figures 2.3 and 2.4.

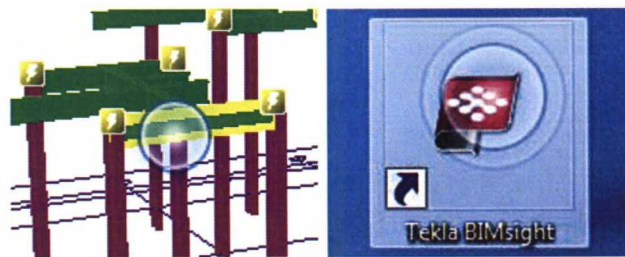


Figure 2.3 & 2.4: Responses in Windows 7 for single and double tapping. Response for a tap is like touching a pool of water where an expanding disc forms.

Perceived performance is not the same measure as actual performance. It can easily be shown that software or even hardware with limited performance may deliver a better perceived performance than software

running on much more powerful system can deliver. This aspect is very much present with the design of a touch interface and thus presents a significant impediment if not done properly.

2.7 Validation methods

There are several validation methods available in usability evaluations. Jakob Nielsen is usually referred to as he presented the ten usability heuristics [54] that encourage for quick and dirty evaluation of a user interface. In addition to these the whole aspect of usability evaluation is encouraged to be a continuing process that takes place throughout the development cycle of a product, instead of being a single step in the lifecycle of the development [55, 57]. For this thesis two methods were specifically chosen to be utilized because of the nature of the evaluation; the System Usability Scale (SUS) [58] and affinity diagramming [57]. The choices were made because of previous good experience with these methods. SUS enables the software to be comparable between different versions of the software, and there are plans to use the score from this study later on. The affinity diagramming is intensive and immersive for the team processing the results, thus making the sharing of information very efficient. The processed results are quite easy to understand and produce a nice mapping of the possibly many findings. The methods will be presented below briefly.

2.7.1. System Usability Scale

To obtain quantitative information about the goodness of the application, SUS questionnaire [58] was used. It is a questionnaire with 10 items which the subjects will fill out according to how much they agree on the questions. The form (in [Appendix A](#)) is scored so that the odd questions are given (*value - 1*) points and even items are given (*5 - value*) points. Results are summed up and multiplied by 2,5 points to reach the overall score which ranges from 25, the worst score imaginable, to 100 and the best score imaginable.

The SUS as an evaluation method has been discussed in more in the study by Bangor et al. [59]. The practical range of the SUS score is that the minimum that would make the product passable being 70 points, and the absolutely great products scoring points over 90. The study indicates that individual evaluations on the SUS score rarely go under 30 points and that the lowest quarter score under 62.26 points [59]. The second lowest quarter scores up to the low 70 in points, second best quarter to the high 70 and the best quartile up to low 90. This means, that a user interface scoring 50 is not 50% worse than an interface scoring 100, but significantly much worse. The SUS scoring is not thoroughly trustworthy if there are a low number of participants, but it is a good indicator of the general level of in-usability for any system. According to the study, the SUS may be used to show that a product is not yet acceptable, but it cannot be used to show that the tested product will be accepted. [59]

2.7.2. Affinity diagramming

The affinity diagramming process was adapted to the contextual inquiry by Holtzblatt et al. [57] in order to make sense from the vast amount of material they had to begin with. Creating the affinity begins with enough notes, observations, critique and other insightful information written on pieces of paper. These paper notes, in the book usually around even 1500 altogether, are placed on a board or table so that those in close affinity to each other are placed close to each other [57]. Each participant will in turn place a new note on the whole and others should then look for more notes that closely resemble that note [57]. The affinity diagram represents the affinity between individual notes in relation to one another and the completed diagram reveals their grouped relationships which reveal hot-spots of trouble in the evaluated system [57].

In this work the method will be used so that each participant will in turn place a new note on the diagram, picking the place with closest affinity to other pieces on the diagram already placed there, and elaborating the chosen affinity a little. This will, in turn, communicate the affinity to all participants and everyone present at the time is

allowed to contribute their opinion to the choice. An important part of creating the affinity diagram was to give meaning to the relationships, and have the persons explain their reasoning when they put a new note in place. Notes on the diagram may be reorganized as pleased when placing new notes, provided that the action is explained to the others.

2.8 Building Information Modeling

Building Information Modeling (BIM) is both a technological approach to construction modeling and a set of processes, designed to support the use of shared technology platform in the architecture, engineering and construction business (AEC). The drawings that are the essence of computer-aided design (CAD) are in BIM produced from modeling the intelligent 3D geometry. The disciplines involved in the BIM process include architectural and conceptual design, modeling the actual design, detailing, erection, fabrication, site management and after delivery building management. It is a life cycle model for buildings, as Figure 2.5 illustrates. It is nowadays the leading way of working in all aspects of the industry, and all of the established technology vendors offer their solutions that implement BIM to different extent. In addition to this there are several other software vendors targeting at building add-on tools for other BIM software. [60]

BIM models are exchanged using the Industry Foundation Classes (IFC) specification. It is a neutral data exchange format specified for the use of AEC business. It has been registered with the International Standardization Organization (ISO) as ISO 16739. The International Alliance for Interoperability (IAI, or buildingSMART International, bSI) is the committee that manages the standardisation process, certifies implementations of the standard and supports the development of the standard. [60, 61]

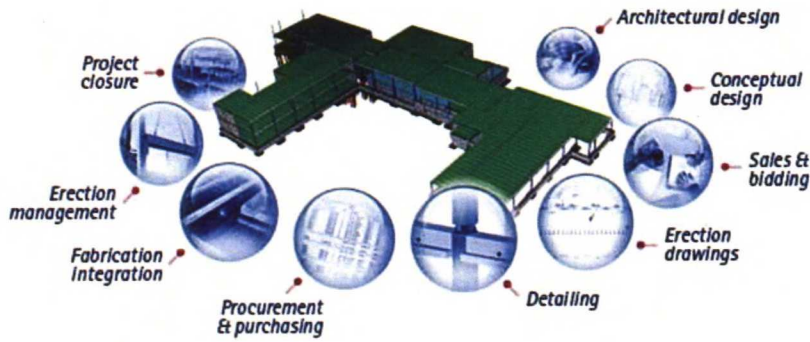


Figure 2.5: Tekla BIM workflow. Copyright Tekla Corporation (www.tekla.com).

The Building Information Modeling (BIM) software that Tekla Corporation specializes in is targeted at the Building & Construction market [5]. Tekla Structures is the main product for that market and its function according to Tekla is to enable “*the creation and management of accurately detailed, highly constructible 3D structural models regardless of material or structural complexity*”. Tekla Structures is capable of producing the IFC model files that may be imported to Tekla BIMsight [5]. There exists several other software available that are able produce this data format [60, 61]. The implementations include IFC2x3 [62] certified commercial products and non-certified free software products and open source projects that are nonetheless capable of handling IFC data model to some level [60].

3. Defining requirements

Tekla BIMsight is an application intended to enable model-based project cooperation between different disciplines of a building project throughout the BIM workflow. Tekla BIMsight enables project stakeholders to identify and solve issues already in the design phase, before construction [63]. It has the capability to display the IFC data model files. Its main features are to review several models at the same time, navigate in them, measure distances, check for visual and computational clashes, and to add notes with markings to illustrate the flaws found from the models [63].

In future Tekla BIMsight will be used on site with a tablet device. This requirement establishes the need to develop support for touch and pen input in Tekla BIMsight user interface. Due to the limitations in usability issues on a tablet device, there was a need to develop support for touch-based user interface in Tekla BIMsight. The target is to build a minimal support for the devices so that the benefits of having a BIM application on site can be demonstrated.

The project was developed using agile methods and requirements gathering was not specifically the implementation team's responsibility. However with this work the sources for the requirements will be explained although listing is omitted.

3.1 Functional requirements

Functional requirements for the development of the touch input interface was as follows:

“Full current and future capabilities of Tekla BIMsight can be used efficiently and effectively on a range of devices including desktop and laptop Windows PCs, Mobile Windows based devices, and future potential to use on popular mobile devices.”

Business segment stated the following as a priority statement to the actual requirements: *“The target is to enable effective use of Tekla BIMsight on the Motion J3500 Tablet.”* Before the project began it was

concluded within the project group and stakeholders of the project, that for Tekla BIMsight to be efficiently usable on a tablet device, especially the target device Motion J3500 [64], it would need to have at least these three features on a usable level to guarantee minimal support:

1. Navigation
2. Measurement
3. UI Layout Improved (optional)

The business segment guided the project team with the implication that the device would mainly be used with a styli in the beginning, but when work with the multi-touch support began and it showed a lot of promise, the focus changed so that the touching was preferred over stylus use. From now on we will concentrate more on the problems faced with the multi-touch features. We assume that stylus usage does not require additional effort from the project team. We also assume that the stylus is the middle ground between using a mouse and touching the display directly, in that it is implementation-wise closer to a mouse as a pointing device.

Usability defines several metrics for defining what efficient use is. Project group has usability experts working with the developers, so attention to the efficient use of the application is constantly evaluated with the modifications. The important findings of this work will be documented in this thesis.

3.2 User stories

In the beginning of the project user interface designer Osmo Tolvanen prepared user stories to support the scoping of specification for the touch support. The four user stories are as follows: [65]

1. As a field engineer, I need to be able to see and navigate through the model while I am walking on site so that I can get a big picture of the situation and answer the questions of my men regarding details of the construction.

2. As a field engineer, I need to be able to use the model for checking measurements that are missing from the drawings my men are using. This happens while I am out on site so that I do not have to always go back to the field office to find the answer, and the walk back to spot to tell the answer.
3. As a field engineer, I add data of the issues found during construction to the project. I take photos and create short notes to be shared with the back office people.
4. As a Project Engineer/Project Manager my company has supplied me a high end tablet computer. I do not have a second computer. I need to carry out full design coordination activity on this tablet.

3.3 Targeted hardware

Tekla BIMsight is targeted at Windows tablets. There are currently a limited number of Windows tablets available. The most notable of those are laptop Hewlett-Packard Elitebook 2760p [66] and tablets Motion J3500 [64], Motion F5V [67], Acer Iconia Tab [68], and the coming Samsung Slate 7 [69]. The main target of the project is Motion J3500 tablet computer that has Wacom Penabled touchscreen that can detect two touch points and a pen.

There are however other touch devices available on the market than just tablets. Since the scope includes pen enabled input as well as the touch based input, most of the graphics tablets such as those produced by Wacom are viable development devices too. Wacom also has devices for desktop environment that can detect touch and pen input. Dell produced a number of computer displays that had an infrared led array on the side. The production has since 2008 [27] been cancelled but the development team had access to one of these devices. Benefit of such device in development work is that developer has access to high-end development computer and is able to debug touch handling code directly on desktop. Lastly HP produces TouchSmart [26] desktop computers which utilize similar infrared array as the Dell display [27]

does and it is a viable replacement for scarcely available tablet computers.

3.4 Main features

Before the requirements gathering had started the latest version that was released to the public internet was Tekla BIMsight 1.3. It was released on October 27th 2011. This version supported mouse navigation only. The main features of the application are listed below. The most relevant features to touch implementation will be covered more closely after the listing. [70]

- Add and remove projects and models
- View, navigate and search in the models
- Save snapshots of the 3D view
- Change colour and visibility of the objects and the models
- Clip the model using dynamic clipping planes
- Check for conflicts visually and with conflict checking tools
- Measure in the models
- Add notes in the projects
- Add relevant project documentation and link it to model objects
- Add mark-ups in the model view to highlight items

The first thing a user need to do, when opening the application, is to either open an existing project or create a new project. The project consists of the models added to that project, their placement, notes, snapshots, documents and detected clashes between the parts. When creating a new project user is presented with a dialog to add new models to it. When a model has been added to a project it will immediately be opened. When a model is opened for the first time it will be cached to the data folder so that the opening process will be faster next time. [70]

The most visible and interactive feature of the application is the navigation in the 3D view. User has six tools available for navigating within the 3D view. [70]

- Orbiting around a point
- Panning the view
- Turning the view around the camera point
- Dynamic zooming
- Zooming into an object
- Rotating the view perpendicular to a surface

These functions are tied to be used with the mouse only. Table 3.1 will list the mapping of the functions to the control buttons. [70]

Table 3.1: Navigation controls

Function	Mapping
Orbit the view around a picked point	Drag with left mouse button
Pan the view	Drag with middle mouse button
Turn the view around the camera point	Drag with right mouse button
Open context menu	Click right mouse button
Dynamic zooming in/out	Roll mouse scroll wheel up/down

Additionally when the user double clicks on a single object in the model, the application will zoom the camera so that the object fits to the view. User may press the Alt-button to modify this behaviour; to turn the camera parallel to the surface they double click upon.

Saved views are snapshots taken of the current situation visible in the 3D view. They retain the view angle, direction, representation, colouring of the parts and mark-ups for later use. User will be able to create new saved views by clicking on the “*Create new view*” button next to the view stripe located at the bottom of the user interface. Saved views are useful for capturing a certain aspect that needs to be shared with peers, or to enable quick access to a certain representation of view angle that displays interesting detail. To open a saved view user will need only to click on the view and the 3D view will be restored to represent that view with a fly-through animation. [70]

Changing the 3D view representation is possible by modifying the representation of all the parts at once or by modifying the colouring of groups of parts as Figures 3.1 and 3.2 display. Using the toggle buttons user is able to switch between solid, transparent and X-ray representation for all the parts at once. Setting the representation for selected parts is done through the context menu for either a single part, or a group of parts. Through the same context menu single parts and groups of parts may also be hidden. Another way to hide parts is using the buttons visible on the toolbar under the View tab above the 3D view. The 3D view may also be switched to represent a perspective view angle or an orthogonal view angle. [70]

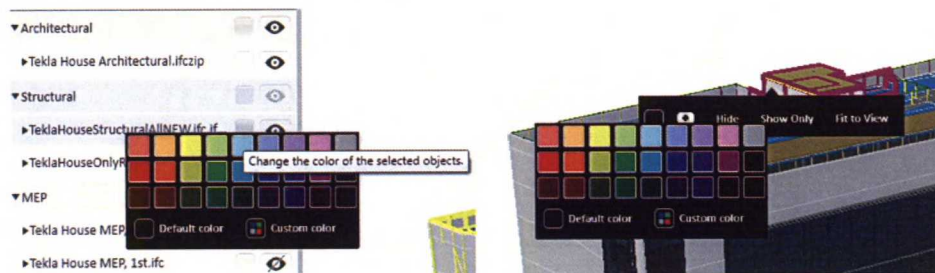


Figure 3.1 & 3.2: Colouring of entire models and selected parts in Tekla BIMsight.

There are different annotations drawn onto the 3D view that supplement information about the visible parts or their relationships. The different types of annotations are flags for detected clashes, documents and notes, redline mark-ups drawn by the user and measurements between the parts.

3.5 Technical background of Tekla BIMsight

Tekla BIMsight is built with Microsoft .NET 4.0 and its graphical user interface is built upon Windows Presentation Foundation (WPF). In WPF the user interface is described using Extensible Application Markup Language (XAML). Microsoft does not specifically encourage one type of use with WPF and there are several customizations with the WPF part of the implementation. Development started out with version 3.0 of .NET but was upgraded to use the latest version because of updates to WPF. Tekla BIMsight is a highly modularized application that has several reusable components. The implementation utilizes so

called application composition framework that is available in .NET through the use of assembly named `Windows.Composition.dll`. Composition enables the high modularization of the application into several separate assemblies that are then combined into the application. The separate components are also highly reusable. It is intended that those components would be reusable in other Tekla software later on. [71]

Tekla uses proprietary in-house 3D technology for its applications. The 3D engine is built on OpenGL and it has C++ bindings. There is a wrapper library for .NET that is in use in Tekla BIMsight and it is enabled for WPF. This library is limited to what functionality has been implemented upon it and it is not directly available for the project group to modify. The project group has circumvented some parts of the interface, to make it more convenient to utilize with Tekla BIMsight. The wrapper did not provide touch events from low level to the higher level application logic layer. This set limits to the project groups for using the 3D component with touch input.

3.6 Technical limitations

Tekla BIMsight has been implemented with Microsoft .NET Framework 4.0 [.NET]. It uses the Windows Presentation Foundation (WPF) for the user interface. Many of the controls are custom made and bear little resemblance to any of the controls provided as part of the SDK. This limitation cannot be changed in any situation and the software has been targeted at Microsoft Windows operating systems. [71]

The implementation of Tekla BIMsight relies heavily on the 3D component and IFC component that are Tekla specific and are installed as plug-ins by the application installer. These libraries provide the low-level functionality upon which the application has been built on. The application architecture is built up from modules that should enable using independent components separate from the other components, but in reality the application requires all the components to be available in order to function properly. The 3D

component has bindings to WPF in order to be embedded into the user interface.

The 3D view is not available to the development team to modify directly. The 3D view is provided as a library and the development team has built additional support for implementing a more modular interaction mechanism for it than what has been implemented in the library.

The interaction mechanisms are called Behaviors and they are controlled by the Manipulator. It takes all input that the 3D view receives, interprets it and passes new events through the interfaces defined in the mechanism into the individual Behaviors, by calling the base class methods coupled for those events. The Behaviors may override the necessary methods in order to have access to the specific input. Behaviors may be modal or not, that enables certain type of a state machine to be constructed with the Behaviors. All the navigational controls and the tools in Tekla BIMsight are implemented in this fashion. [71]

This architecture and the 3D component were designed to support mouse and keyboard input only and touch input was not handled on any level. Raw touch and stylus input was not passed over to the component as the component was not subscribed to the raw input events. The 3D component uses OpenGL to render the 3D view onto the control region. The WPF bound control area region is bound to the OpenGL process using the HWND handle of that control in order to have the interoperability between the two technologies over that region of screen. In practice another application window is created to match and follow the specified region and OpenGL then has the ownership of that region of the screen when it is on top. This Win32 and WPF application interoperability comes with a technical limitation that is referred to as the airspace violation [72]. WPF controls hosted within the same window cannot be rendered on top of the area owned by the Win32 application and the Win32 application cannot render on other regions than what it has ownership of.

The airspace violation also limits the ability to receive input as the input processing mechanism of Windows is built to handle the regions that are bound to the technologies. Suppose that the Win32 region is hosted on top of the WPF region. If the mouse cursor is dragged from the WPF region over the Win32 region, it appears to the application code as if mouse had left the region of the WPF such as when the mouse cursor leaves the window region. If the mouse input is forced to be captured in the WPF control then the Win32 will never receive the input. This in turn would disable the input handling that has been implemented into the 3D component used by Tekla BIMsight.

It is possible to use Tekla BIMsight version 1.3 with touch-enabled devices or with stylus to a certain extent. Stylus input may be emulated by the drivers in Windows versions previous to Windows 7. This support depends entirely on the driver implementation and it is not possible to gain any support for the application other than what has been configured to the drivers. Windows 7 is the first Microsoft operating system to support touch input and it has the possibility to produce an emulated input too. Touch input may be configured so that it acts as a mouse device and the programs that have not implemented any level of touch support may still be used — to a certain extent. Table 3.2 describes how different gestures have been mapped in the emulation mode of the touch input handling.

Table 3.2: Touch input emulated as mouse input.

Touch input	Emulated event
Single finger tap	Left mouse button click
Press & Hold with single finger	Right mouse click
Pinching with two fingers	Mouse scroll wheel
Single finger drag	Dragging with left mouse button

The technical limitations may be summarized as following:

- Mandatory use of .NET Framework 4.0.
- Mandatory use of Windows Presentation Foundation.
- No direct access available to touch input through the 3D view.
- No direct access to the 3D component in order to implement the support.
- Possibility to subscribe to the application programming interface (API) of Touch API in .NET Framework when operating system supports this.
- Airspace problems with WPF and Win32 interoperability.
- Input handling of the 3D component.

3.7 Requirements for the UI

The user interface was already at an early stage identified to be problematic for touch interaction and a proposal of an improved interface was made. The full refactoring effort of the UI was not invested among the prototypes, but rather pushed back to a later time. Herein are listed the problems and requirements that were identified from the user interface at this time.

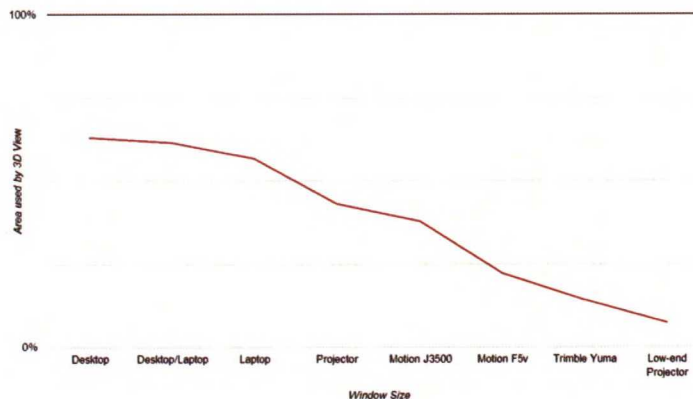


Figure 3.3: Screen estate available to the 3D view in different devices.

3D view is the most important control area on the application. On a small screen device the UI suffers greatly from the fact that the 3D view area is not maximizable. There are always some controls eating up the space and calculations were made to elaborate how much screen estate is available to the 3D view. Estimate of the available screen area,

illustrated in Figure 3.3, was made by estimating the constant area that different controls were using by default and reducing that from the screen area in different resolutions when the window would be maximized.

The UI needs to be such that it meets with the minimum requirements to support touching. According to the Microsoft Developer Network (MSDN) documentation [2] applications for Microsoft Windows are considered to be:

Touchable when [2]:

1. The program's interactive controls are at least 23x23 pixels (13x13 DLUs).
2. The program has good keyboard and mouse support for relevant system gestures such as flicks, multitouch gestures, and drag-and-drop are functional.
3. No tasks require using hover or the touch pointer.
4. All controls use Microsoft Active Accessibility (MSAA) to provide programmatic access to the UI for assistive technologies.

Touch-enabled when [2]:

5. The most frequently used controls are at least 40x40 pixels (23x22 DLUs).
6. Relevant gestures are supported (including panning, zoom, rotate, two-finger tap, press and tap), and the effect occurs at the point of contact.
7. The program provides smooth, responsive visual feedback while panning, zooming, and rotating so that it feels highly interactive.

Touch-optimized when [2]:

8. Tasks are designed for easy touch by placing the most frequently performed commands directly on the UI or content instead of in drop-down menus.
9. The program's special experiences are designed to have an immersive touch experience (possibly using raw touch input data), with multi-touch manipulations and details like having feedback with real-world physical properties, such as momentum and friction.
10. Tasks are forgiving, allowing users to correct mistakes easily and handle inaccuracy with touching and dragging.
11. Tasks are designed to avoid or reduce the need for heavy text input or precise selection.

To reach the minimum level of touchable application support the application would need to fulfil the requirements 1-4 mentioned above. Of those the requirements 3 and 4 will require attention because neither is yet met.

To reach the second level support of being touch-enabled, attention would need to be put to all requirements 5-7. Buttons are currently small, there is no gesture support and performance is unknown.

To reach the last level so that the application would be touch-optimized, requirements 8-11 would need to be met and none of them are met currently.

4. Implementation

There were several areas that required attention in the application to make it fully touchable like Microsoft documentation suggests [2]. Places that need attention are illustrated in Figure 4.1. The elements in the UI are lists of all kinds (in orange), buttons and their size (in pink) and the 3D navigation (red) that is based now on mouse input. A series of prototypes were created for the touch input system and this chapter will mainly concentrate on them. The prototypes covered the angle of attempting to go around the problems and then finally solving it after all. The original intent was to produce the minimal necessary support for touch input devices and not a full compatibility.

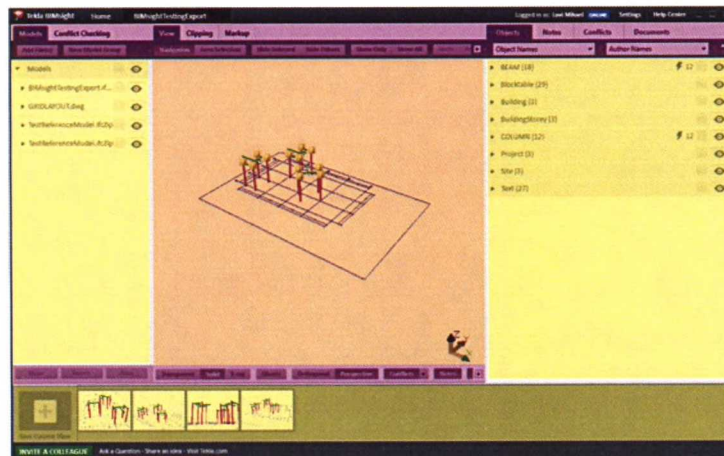


Figure 4.1: Areas in the application UI that require attention for touch support.

The effort of finding the information necessary for the touch implementation was taken on by two persons in the development team and one person in the technology unit. The person in technology unit is also responsible for the implementation of the 3D component and he was looking to build the low level support for touch input directly.

4.1 Iterations of touch support

The implementation was done in several iterations or sprints over a period of two months. During the development period problems were solved as they arose and the design of the solution changed many

times. Five sprints passed during that time that roughly coincided with the prototypes of functionality implemented. The prototypes overlapped each other sometimes and most of the time they were passed over from one sprint to another. The different prototype implementations may be summarized as follows:

1. 3D Technology component as a source of touch events
2. Emulating mouse events with touch events
3. Using both WPF Manipulation and WPF raw touch events
4. Using only WPF raw touch events
5. Performance optimization iteration

First sprint goal was to “*Enable 3d navigation with pen*”. By this time the assumption was still to emphasize the stylus control over touch control, as it was assumed to be easier. While the team would investigate possibilities for going around the technical limitations in other ways, it was known at the time that the technology unit would be implementing the low level events support and we would have access to that by the end of the sprint.

In the beginning of the sprint it was also acknowledged, that we would not be completely done with the new navigational system and that we would need to refine it. It was written down as following: “*Since there is no specification available we expect that we will need to fine tune these controls in next sprint according to feedback.*” The goal was also to investigate many possible solutions for the navigational controls, as we assumed the user interface to be of great importance to the users.

Part of the reasons for choosing the sprint goal was that we attempted to go around the limitations mentioned in previous chapter, such as that we did not have the possibility to subscribe to low level events through the 3D component, nor could we use WPF events because of the airspace problems. We then chose the obvious route for us and we attempted to go around the problem by enabling the navigational controls for the user, by providing them a user interface with which to accomplish most of the things they would need. The goal became then to *create control buttons for navigation.*

4.1.1. 1st prototype: Joystick buttons

Our main goal was to produce testable control buttons for navigation in Sprint 1. We were trying to fill the use story: *“As a user I want to be able to navigate in the model using a pen, a touch-enabled device or a mouse”*. Better navigation was tested first with virtual joystick buttons. First testable version of this functionality did not feel *‘fluent’* and many who tested it did not see the value of the solution.

Our proposal was to create a digitally simulated analogue joystick buttons, which the user could continuously use to control their navigation in the model. These controls would be a joystick for each action separately; rotation, zooming and panning. Rotation would require two dimensions to work properly, zooming functionality one dimension and panning also two dimensions to function.

Rotation and panning buttons would be divided into 4 regions to get the user's attention to the idea that the control could be used to affect change in those directions. In the center of the control there would be a fifth button called the 'knob' that represents the virtual joystick. By dragging the knob towards the edges of the control user is modifying the speed at which the 3D view is changed. Zooming is limited to vertical directions only because it only requires one-dimensional changes. Hence the distance of the knob from the centre position equals the speed of change.

The controls would have other ways of using them in addition to the knob. User may press any of the four regions of the control to modify the 3D view by steps. Rotating and panning will then have 4 button areas, whereas zooming will have 2 areas.

Optionally, when user presses the knob itself this should toggle the corresponding behaviour to become active for the left mouse drag on the 3D view. This doubles as the default tool behaviour already present in the toolbar buttons.

The navigation was first implemented so that every 150 ms the location of the knob was checked and another animation sequence to fulfil that

alteration was initiated. It was recognised that this should be done so that the animation is continuous and that it feels fluent instead of ‘chunky’ as it felt with the first attempt. To express it otherwise, the animation itself was fluent, but unresponsive to user actions.

Improved version of this was to attempt to draw whenever possible — slower machines when they can and faster machines more often — and upon each redraw get the passed time since last redraw and affect the change needed by

$$position = speed * time_{passed}$$

Instead of just affecting some change every time a redraw occurs. This yielded much more consistent behaviour for the joysticks *continual change*.

The joystick buttons were achieved with an overlay window that adapts to the changes of the main window that is the parent control of the overlay. This is described by its creator as following: “Adding `FloatingControl` creates a new styles and transparent window that follows the location of parent control of `FloatingControl`. If sizing binding of window is needed, it can be done in XAML code that creates the control, since window resizes to its content.” Below is a sample of how the overlay window is defined:

```
<Grid x:Name="ModelViewGrid" >
  <ContentPresenter
    Content="{TemplateBinding Viewport3D}" AllowDrop="True" />
    WindowVerticalOffset="0"
    WindowHorizontalOffset="0" >
    <Border
      Background="Transparent"
      Height="{Binding ElementName=ModelViewGrid, Path=ActualHeight}"
      Width="{Binding ElementName=ModelViewGrid, Path=ActualWidth}">

      Content of the overlay window is placed here

    </Border>
  </WPF:FloatingControl>
</ContentPresenter>
</Grid>
```

The overlay window still required a lot of finalizing work, so it was continued in the following prototypes. More support would be implemented according to needs.

This approach partly overcomes the airspace problem and enables any controls to be drawn correctly with alpha transparency and capture all the events from that display region. This was completely opposite to the documentation in MSDN [73], but it is understandable considering that this approach only works with Windows Vista and later versions. This presents a new technical limitation to the software, as Windows XP is not supported by this solution. Windows XP still has a large user base with Tekla BIMsight and they will not be able to use any controls implemented with this approach. Therefore it was decided that only those controls vitally requiring this approach would use it.

4.1.2. 2nd prototype: Touch event translation to mouse events

The sprint goal set by the team stated that we would “*Implement the chosen navigation controls.*” However, throughout the sprint the final decision for the choice of the navigational controls was delayed.

When the newly implemented support of receiving low level events through the 3D components interfaces became available, the team started to investigate how to utilise them. The goal was laid out to “*Try the design possibilities provided by 3D-component multitouch support*”.

The first mechanism that was thought out was a mechanism that would utilise the existing navigational mechanisms to the extent that would be easy to implement in order to achieve minimal support. Already at the time, there was an idea that it might be necessary in the long run to implement a proper system, but it was chosen that we would rather first try out a simple solution. This solution interpreted the touch input provided by the 3D component interfaces into mouse events consumed by the navigation behaviours described in Chapter 3.5.

The mechanism would rely on toggle buttons that would be visible on the user interface which would enable user to switch between different behaviours or tools. These behaviours were to be orbiting, turning, panning and zooming at first. This mechanism was built into the

behaviour mechanism so that users had a full control over the tool that they were using with the mouse buttons. The second mechanism was the touch input interpreter that was working alongside the behaviour mechanism, feeding it with the events coming from the low level touch interfaces. Together they enabled a rudimentary touch control over the 3D component where the touch inputs were coming from the 3D component directly.

When the first functional prototype was working, we got to try it out and the solution looked very promising. We then started to experiment with variations on the mechanism and the behaviours. The tools offered for the user actually have certain behaviours corresponding to the action of pressing one finger down, and other behaviours for when the user is pressing down two fingers.

The functions that modified the camera within the previously listed behaviours were also experimented on to create new behaviours. One such behaviour was walking in the model when the vertical movement of one's finger changed the zoom factor of the camera and the horizontal movement of the finger was interpreted as camera translation sideways; or panning of the camera (sideways only). Later on this experimentation proved useful and the panning was modified to turning the camera sideways (yaw) and the behaviour was named walking, because the camera was also set to follow any surface underneath it. Other observations of the implementation were as follow:

- When walking forward, finger movements should reflect the resulted direction. Up moves the camera forward, but moving the finger left yields turning right.
- Changing the mode between walking and orbiting when user selects some parts.
- The variables used to determine exact behaviour of walking need a little bit of work and the '*clutchy*' movement should be fluent.

When the behaviours were given for others to test we received feedback that the behaviours of the camera felt odd. The result of the discussion was that the way rotation was working was not good enough. It followed the paradigm that when *user drags the point on the screen as if they were dragging the camera lens to some direction*, as illustrated on left side in the Figure 4.2. All other actions followed the assumption that user picks a point and that point stays under the finger. This needed to change. The solution was to change the paradigm so that the *user was dragging a point in the 3D world in the direction of the finger movement*, as illustrated in Figure 4.2 on the right side. This allows for a better connection with the model and the real world. The end result of both actions is illustrated by the changed image of the 3D view underneath each.

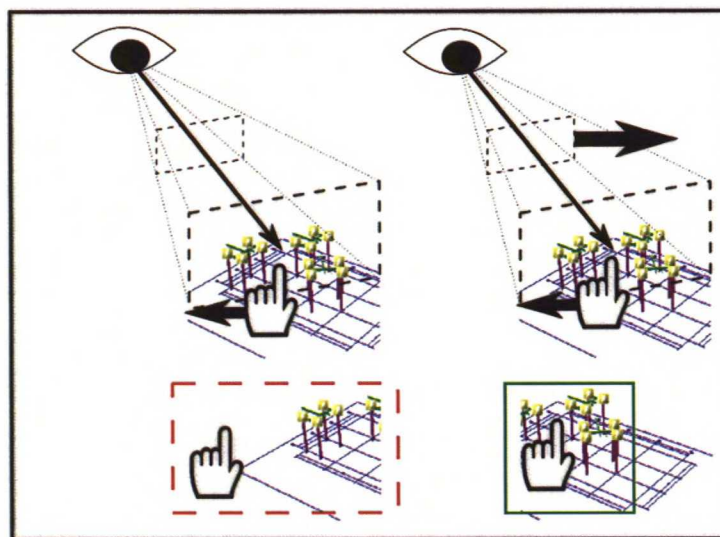


Figure 4.2: Graphical comparison of the turning behaviours. The left side shows the old way and right side the new way and lower image of a model is the final result in both.

The good thing about emulating mouse movements with the touch input was that it utilised existing framework of the behaviours. There was a serious flaw with the approach though, and that was in the interpretation of the events. We had serious difficulties in interpreting the events just right when the amount of fingers was changed in the middle of input. This was especially highlighted, because the mouse navigation was relying on modal input events in a way. That is, when one button was pressed down and mouse had been dragged, the action

of pressing down the mouse button ends with the releasing of the same button. Emulating this behaviour correctly was nearly impossible.

Touching with a finger triggered a `MouseMoved` event from the mouse driver. It could be that other mouse events were leaking as well and we did not check for them. Furthermore after last touch event had indicated that the finger had been lifted, `MouseMoved` events would be received with parameters indicating mouse left button was still being pressed down. This caused behaviours to work incorrectly.

In the prototype all mouse event were turned off permanently when first touch event was received. A flag in the 3D component `IsMultitouchFound` was supposed to tell if multitouch device was present, but that information was not reliable.

It was observed that it was a lot easier for users to produce unwanted actions with fingers than by using a mouse. For example:

- Tapping of two fingers (aka. right mouse button click) surprisingly easily produced a double click.
- Double tapping to different points in screen was easy and natural to do, which means location checks must be done within the timeframe of the taps. In practice a user might tap their fingers from little finger to index finger in rapid motion.
- When dragging by touch it was easy to:
 - Accidentally touch the screen with another part of the hand, especially when using multiple fingers.
 - Accidentally lift a finger from screen when moving can happen occasionally as well.

- Flicking (touch+move+lift) could be done faster by touching than with a mouse. Flicking was also more natural to do with a finger than the mouse. This became obvious with timers in behaviours like orbiting and area selection when:
 - Touch started the area selection timer,
 - Then by moving the cursor the modal orbit was started,
 - Then lifting the finger stopped the modal orbit, which broke the event chain without telling it to the area selection behaviour and finally,
 - The timer for area selection fired and that behaviour was activated.

The two persons working on the support believed the problems could be resolved, but the rest of the team decided to push for another solution that would be more maintainable later on. Furthermore the lesson from this mechanism was that it is very difficult and mostly not feasible to attempt to emulate another input device. This mechanism was later on fully removed from the application code base.

4.1.3. 3rd prototype: Full multi-touch implementation

For this next prototype a full touch support would be implemented according to the interfaces provided in the WPF. This support is would be built using the Manipulation interface [73] in the framework.

The prerequisite for implementing this mechanism and for at all catching the events produced by the framework was the transparent floating window on top of the 3D component. It was realised that if we are able to draw transparent controls on top of the 3D component that are able to receive input directly, then maybe we could subscribe to the events being transmitted to the window by the framework and proxy them back to the parent window. The way to proxy the events to the proper handler was done through event proxy that works through two subscription methods in the `WorkspaceManipulator`:

```
public void AttachInputProvider(Control control)
public void DetachInputProvider(Control control)
```

The attaching method subscribes the manipulator to the relevant events through the specified Control instance that already has the necessary events as public members. The detachment method is there to provide a safe destruction mechanism. Events for single finger movements are passed on as raw touch events, while multi-fingered events are handled as manipulation events. The relevant manipulation events are as follows:

```
TouchDown
TouchMove
TouchUp
ManipulationStarting
ManipulationInertiaStarting
ManipulationStarted
ManipulationDelta
ManipulationCompleted
```

The behaviour mechanism that is integral part of the navigation mechanism required changes in order to incorporate new messaging event in it. An event called `ManipulationChanged` method was introduced in the behaviour handler interfaces. The virtual method consists of the following parts:

```
Response ManipulationChanged(
    IWorkspace workspace,
    Vector2 origin,
    ManipulationDelta delta,
    ManipulationVelocities velocities) {...}
```

This mechanism passes just the necessary information about the manipulation event that has just happened and nothing more. This approach is good enough if you do not want to know anything more about the actions that take place with fingers. The shortcoming with this approach was that it was very difficult to find the original Manipulator instances responsible for the change and to utilise that in the behaviours. Another shortcoming was that the performance appeared to be really bad. Another good thing about using this ready built mechanism was that it had a simple implementation of a physics engine to simulate inertia for finger gestures. Other notes from the `ManipulationChanged` event handling were following:

- Better to perform simple operations with repetition than complicated tests
- If inertia is available it should be used, because it increases the amount of calls and smoothes the gestures.
- When using inertia, remember to compensate for the "loss of a finger", that is when a finger is lost from Manipulators.

One very handy operation that was done during the processing of the ManipulationDelta event was to compensate for the loss of a finger input. For the kind of behaviour that was used in multi-fingered operation we did not have a special need to maintain the touch points and to use that information in the behaviour. The behaviour mechanism works so that the modal behaviour that is active will not stop until all fingers are released. Rather we would observe an odd twitching when a finger was released during the two fingered pan behaviour. It was possible to fix this problem by monitoring the center position of the fingers, then when a finger was lost from the input to add a compensating vector to the input position so that the behaviour would continue smoothly. Example below is the compensation mechanism, where e is the event parameter:

```
if (manipulators.Count == e.Manipulators.Count())
{
    positionOffset = new Vector2();
}
else if (positionOffset.Length.Near(0.0))
{
    positionOffset = previousPosition - position;
}
```

We also found out by reading the documentation [73] further, that the interpretation from the raw touch events into the Manipulation events was made in another process through and not in managed code. The other process was an unmanaged process and it was called with Remote Procedure Calls (RPC). It occasionally got delayed and was waited for until the RPC timeout cancelled the waiting. It is a nice possibility that Microsoft decided to implement the Manipulation events, but we felt that we needed to do better than that. The interpreter for the WPF Manipulation events was built into the external window and made to interpret the manipulation events into actual changes in the 3D camera.

This prototype had performance problems with most gestures and it always felt like the interaction was lagging behind badly. Pinch zooming and panning with two fingers were especially bad with manipulations. Another thing with the approach was that it had to combine single touch input and multi touch input in a way that was awkward to handle in the tools that finally implement the behaviours. The good thing was that the inertia was working well and made coping a little easier. The conclusion in the end was that the dependency on manipulation events had to go. It was not fulfilling our needs properly.

4.1.4. 4th prototype: Using only WPF raw touch events

In this prototype the combination of multi and single touch events was merged in order to achieve more easily maintainable code base and to simplify the structure. Another hope was that if we can cut down on the time it takes to handle the events, we could improve the performance.

Testing the previous implementation gave us a good view into the performance directly when using the software, it was not good enough. It seemed as input was being handled a second later than it should have affected the model view. This gave us a hint that there might be something wrong in the handling of the events. Further testing revealed us that the events being pushed from the driver were not handled at the same speed at which they were delivered, so that there were always more events in the queue than what was being handled. This was not clear to discover as we had already had some problems interpreting how the events arrived to the event handler methods in the first place, because with manipulation events we ended up sometimes receiving a `TouchMove` event after `TouchUp` event. Further study revealed that in fact the events were in a sense being pushed to the application and that there really were always more events queued up than what the handler could cope with. The handler methods were taking too long to execute and we needed to change that. The bottleneck was especially in handling the `TouchMove` events.

The solution that was devised for this was to have the handler methods `OnTouchDown`, `OnTouchMove` and `OnTouchUp` only catch the events

and mark them down for processing later on. The methods sanity check the input before handling it further and make certain, that the input provider captures the device feeding the input so that the whole screen may be used during an single input action. The events arrive and are forwarded as following:

1. Event `TouchDown` is received and the manipulator involved is registered as a followed source for `TouchMove` events. All previous `TouchMove` events need to be dispatched before registering additional sources. Objects and geometry under the manipulator are updated.
2. Event `TouchMove` is received. Input is sanity checked against the registered manipulation sources. The event is handled only if the manipulation source is registered. Next duplicate inputs are filtered out by comparing the previously registered touch position. The input is stored to be handled later. If a timeout of 1/25 seconds has passed a call to method `DispatchPendingTouchMove` is invoked to be raised with a dispatcher queue priority for rendering items.
3. Event `TouchUp` is received. Input is sanity checked to be from a touch device. All pending `TouchMove` events are dispatched. If the source for the event is registered then it will be unregistered and device capture released.
4. When `DispatchPendingTouchMove` is called the timestamp of the dispatch is recorded and the behaviour mechanisms `TouchMove` handler methods are invoked.

There is by default only one such handler instance in the behaviour stack that accepts the input from the 4th step described above. That is in the `TouchNavigation` class that is responsible for initiating the proper behaviours to handle the input correctly. It takes care of identifying gestures from the given input so that they might more easily be used in the actual behaviour implementations.

The gesture recognition is built into the `ManipulationProcessor` class that is easy to extend to recognize more gestures. Currently the

recognition only distinguishes between single-fingered operation and multi-fingered operation. When multiple fingers are used the recognition dot product is calculated between the movement vectors of the fingers to distinguish between gestures. Movement vectors are determined by subtracting the initial position of the finger from the last position. The resulting movement vectors are normalised. Dot product is calculated between the normalised vectors and then interpreted. Below the function is explored in two-finger gestures of panning, pinching and rotating. Only the first two scenarios were implemented and are in use.

$$dp = v_1 \bullet v_2 = v_1^i * v_2^i + v_1^j * v_2^j$$


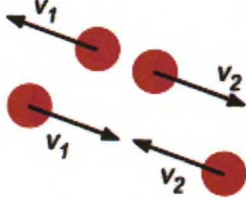
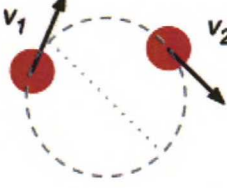
Pan	Pinch out and in	Rotating
		
$1 > dp > tol$ $tol > 0$	$-tol > dp > -1$ $-tol < 0$	$tol > dp > -tol$

Figure 4.3: Illustration of different gestures and how the dot product distinguishes them.

Detecting the rotating movement of the fingers is difficult with this detection method. In the Figure 4.3 the circle drawn over the fingers is the trajectory of the fingers, when the tangents of the fingers movements set circle. If the fingers are exactly at the opposite sides of the circle, the detection mechanism would read the movement as a pinching gesture. Identifying the rotation movement of the fingers reliably would require identifying the circular trajectory to some center point. This was not implemented and the value of the dot product would be the same with pinching if used directly. When the amount of fingers was changed the behaviour would also be changed

immediately. In future we wish to be able to utilise the geometric points under the fingers of the users.

`TouchMove` handler in `TouchNavigation` calls the `ManipulationProcessor` to process the input. As a result a gesture is returned. The result details if the input has been single-fingered or multi-fingered action and in case of multi-touch it gives information if the gesture was pan or pinch. The mechanism identifies the tool in use in the navigation and instantiates the corresponding behaviour into the behavioral stack. Constructor injection is used to pass the singleton instance of the `ManipulationProcessor` to the behaviours to allow them access to the processed input. The singleton is maintained in `TouchNavigation` class.

After the implementation was finished the performance was again evaluated with the team members testing the software on the tablet devices and the desktop displays available. The performance for interpreting the touch input had already increased significantly and now the system was able to cope with the finger movements with relatively less delay than previously. It was obvious to the people who had used the system before and after that there was a delay when comparing the performance of the mouse input to touch input. And the issue really was somewhere there still. It made sense to assume that since the mouse based input is working so well, why should the touch input handling not work as or close to as fluently?

4.1.5. 5th prototype: Performance optimization iteration

To solve the remaining performance problems their more precise location had to first be pinpointed. With the 5th and final round of implementation the target was to get rid of the rest of the problems in the touch input system so that it would at least be reliable in use even if the performance was not yet on acceptable level.

There were problems remaining with the touch input system still, as a bug description entered after finishing the 4th prototype states: *“WPF UI is not responsive on touch, but 3d view navigation still works.”*

This specific problem was solved by disabling manipulation events in the overlay window after which the problem would no longer recur. There were small problems with the UI, renewed snapshots features, architecture and maintainability issues and fine tuning of the gesture recognitions.

The performance of the touch navigation had been poor regardless of several attempts at improving it. The touch input event system was fast enough to process all the events, but there was a certain bottleneck elsewhere.

A performance test was laid out for finding out the exact spot of the problem for achieving higher rendered frames per second rate. The emphasis was to look at the methods that directly trigger the rendering activity in the software. There is only a single class that possess the feature to manipulate the 3D camera properties and that is in `WorkspaceCamera`. It was then logical to try to trace those methods that call the `CopyFrom` method from that class. There are few locations that use this method:

1. Camera animation behaviours
2. Incremental navigation buttons
3. Presentation manipulation
4. Loading of snapshots
5. Navigation input behaviours for mouse and touch together

The most interesting of these spots is the 5th line and, for example, for orbiting behaviour the usage of a method called `UpdateCamera`. This method is called in methods `TouchMove` and `MouseMoved` which are the callbacks for touch input mechanism and mouse input mechanisms respectively. These methods were monitored in the performance testing tool called `JetBrains dotTrace` with evaluation license. The measurement tool works so that first the application is launched and preliminary launch related things may be executed manually in the program being evaluated. After that the capturing for performance measurements is initiated to collect the actual data. Data was only collected for certain duration of time in which the application was

saturated with the respective input and then shut down immediately as the data collection ends too.

The test for this defect was conducted on the Motion Tablet J3500 with about 1900 objects visible. Part of this study was to take a look at the JetBrains performance profiler, dotTrace, with Tekla BIMsight. Results were staggering.

Table 4.1: Test durations and numbers of calls to respective update methods.

Device	Test duration	Number of calls
Mouse	22 s	628
Touch	25 s	68

The Table 4.1 above presents the results of the performance results and it is interesting to find that in fact even if the touch input handling mechanism is able to process hundreds or thousands of input events during this test run, only less than hundred calls to modify the 3D view are made. If we assume that the numbers may be directly translated into frames rendered per second (fps) these yield 28 fps or mouse input and <3 fps for touch input. These numbers reflect closely the experience that the touch input in the software feels like in action.

The original problem was in the fact that there were many touch events coming through and they saturate the pipeline where the rendering calls are being made, that being the very same pipeline. The solution to fix this was to build a sort of double buffer for the touch input events and then send another event to the same pipeline to trigger the processing of the past move events which is saturated already as stated.

The working solution is to use the same approach on the workspace camera, which directly triggers the 3D view to render. All modifications to `WorkspaceCamera` no longer directly trigger the rendering on the 3D view, but rather just push modifications to a transaction in the `WorkspaceCamera` that will be committed to the 3D view when the `Rendering` event is triggered for the

WorkspaceManipulator. That event gets triggered by WPF every time the UI needs to be redrawn.

Second part of the solution is to trigger the processing of the input events from the first buffer sparsely enough for the gesture recognition to function with the input, and so that it has enough input to process, and yet often enough so that the touch interaction does not feel sluggish to the user. This happens so that the call is not invoked until:

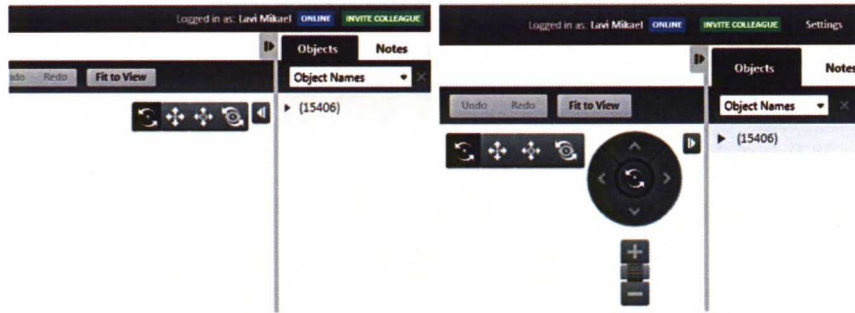
1. A certain time period has passed to allow the input events to be gathered (1/25 s)
2. The previous dispatcher object has been disposed of
3. And the call gets invoked in the dispatcher queue using the rendering priority so that it is in sync with rendering.

4.2 Delivered functionality of the Tekla BIMsight 1.4

All the features specified earlier were delivered when the version 1.4 of Tekla BIMsight was released. The new features that were highlighted in the release notes of the version 1.4 [74, 75] were the following:

- View groups as major new functionality
- Slide shows as major new functionality
- Touch controls as usability enhancements
- Possibility to hide and show navigation controls
- Ability to switch for the Tablet user interface mode

The user interface was reviewed and tested thoroughly during the development period to have all the UI elements respond to touch. Some areas required more attention as explained earlier, but other places work with the out-of-the-box functionality that the .NET framework provides. Popup controls such as the colour picker and part information popup were made to support touch input. All the lists were made scrollable with touch. All critical controls in the UI may be toggled to become larger for tablet devices using the Tablet user interface mode. The differences are illustrated in Figures 4.4 and 4.5.



*Figure 4.4 & 4.5: Normal and Tablet modes for the user interface.
Both images were captured from the exact same area.*

The 3D view is fully controllable using either mouse or fingers or stylus. There are additional navigational controls available to support stylus users, but they may also benefit users of fingers and mouse as well. These controls are located above the 3D view and at the top right corner of 3D view area. Performance of the 3D is only limited by the hardware and the resulting performance should be equivalent on any of the input mechanisms.

5. Usability evaluation

The system was usability tested after the implementation to evaluate the goodness of the choices made for the implemented system. Methods used in the usability evaluation have been described in more detail in section 2.7.

The user testing was designed to evaluate the usefulness of Tekla BIMsight on a construction site. The evaluation was a walk-through session with an expert on BIM usage on site and non-formal interview. The evaluation was conducted using standard usability testing methods by meeting with users and observing their activities and interviewing them. Test session featured the filling of System Usability Scale questionnaire (in [Appendix A](#)). For analysing the material gathered from the sessions an affinity diagram was created later after the session.

The test session was organized by Taru Lääkkö on behalf of Tekla. She was the test participants primary contact person. She instructed the users how they needed to prepare for the evaluation sessions and provided them the necessary information.

Prior to the visit the evaluation team will review their equipment according to the predetermined checklist (in [Appendix B](#)), so that all the required equipment and tools are not forgotten. The most important things were:

- Consent forms for material use rights
- Copies of the SUS forms for the test participants
- Checklist for the instructor that contains all steps in each phase
- Help reference material for the user
- Video camera for recording the sessions, along with charger and tripod.

5.1 Planning the evaluation

The design of the usability evaluation started from the need to understand the way of working with a tablet computer on a construction site by a subject whose main task is coordinating the tasks

there with other people. The subjects would most likely be working in an office on site, but should also have an occasional need to go to the construction to inspect the work first hand. The main target of the evaluation was to understand what that need was, coordinate the visit beforehand and the activities on the visit. The visit was arranged on February 7th, 2012.

The people present on the visit were instructor, camera operator and two observers. Roles were appointed at the briefing meeting on the day before the evaluation session. People attending the visit as the evaluation team were Mikael Lavi, Taru Lääkkö, Osmo Tolvanen and Marko Myllymaa. The evaluation session was been designed by Mikael Lavi with assistance of Taru Lääkkö.

The evaluation was conducted in three phases; first in the construction site office, then on the site with the expert using the Motion J3500 tablet device and finally back at the office for SUS and feedback. The research objectives for these sessions were the following and the team was looking for qualitative answers for them.

1. What is the main method of navigating in the model and how will they adapt it with the tablet device?
2. How quickly would the users be able to adopt the touch-enabled navigation controls?
3. What is the reception of the measuring tool using touch?
4. What is the user's preference for the input mechanism; touch, styli or neither? Why?

The first objective was targeted to provide us with feedback on how to instruct the user better with the touch navigation. By understanding how they adapt their existing navigational behaviour to the current implementation, we could maybe provide more insightful guidance to other users on repeating the adaptation. We were more specifically interested on their primary navigational tool and were looking for that information.

For example one supposed model of an efficient navigational behaviour is to have the '*Turn tool*' active for one finger dragging and

rely on the pinch zooming and panotate gestures when navigating inside a building. Outside the building we have observed that the Orbit tool is more useful.

5.1.1. Evaluation goals and participants

The evaluation was conducted on Skanska Oy construction site. There were two test participants. Before the visit to the site, the subject was asked to pick a few targets for inspection on site. These points were supposed to represent real targets for inspection, and to reflect actual usage scenario for the test subjects. One participant was required to prepare the model files necessary for the evaluation session and asked to prepare a project in Tekla BIMsight that contains the relevant models. Additionally it was requested that the project would have been filtered to represent that area where the inspection will take place. The users are not assumed to know how to operate the software before the evaluation.

User #1 is a development engineer from Skanska BIM Competence Center and an expert on BIM and its usage on the site as well. User #2 had a similar background and he was responsible for picking out the inspection points on site. He has previously been using Tekla BIMsight and his tasks in the BIM Competence Center include evaluating new ways of using this software on site. He has also taken part in a previous usability evaluation session in Tekla headquarters with others when the software was still in development.

Upon arriving in the construction office, the test subjects need to give their consent (in [Appendix C](#)) for the recorded material to be used in evaluation purposes by Tekla and in this thesis work. Consent from the users was asked for snapshots of the video material, mention of the company and mention of persons participating in the evaluation. All sessions were recorded using a handheld video camera capable of recording 1080p video footage. Next the instructor of the session gave a quick overview of the plan for the session. This was done to give the subjects rough idea of what to expect.

5.1.2. Limitations

The evaluation was focused in this evaluation to the navigation and measurement tools. Users feedback was gathered at all times if they encounter difficulties, but to keep the focus in predetermined areas users were instructed not to dwell too much in the problems.

Another limitation that the instructor was to tell the users, was the limitation that the user cannot measure distance between two planes. This is technically limited out from the functionality of the tool.

The instructor asked the participants to think aloud when they were using the software, and to elaborate as much as possible. They were encouraged to think aloud and explain their thoughts as they explore the user interface and do their round. The evaluators were also asking relevant questions when such came up.

5.2 Execution of the evaluation

5.2.1. Inspection briefing

The aim of the inspection briefing was to understand how the users were using Tekla BIMsight when navigating the model, marking views down to return to them later on, and how they used notes. The briefing went so that User #1 was asked to present, using Tekla BIMsight, where he had planned to take the evaluation team visiting on the inspection round. He was instructed to do this using the Motion J3500 in the laptop mode, with the docking station and a mouse, to accustom him to the device in familiar context. During this briefing the user should be encouraged to create the views that were supposed to be used on the inspection round. By creating the views in the new version of the software, the user was immersed into the software more.

When the briefing was done and all preparations for the inspection round were taken care of the users were shown how to use the touch features on the tablet computers. The Motion J3500 device was separated from the docking station and the mouse for this. The user was then given the documentation as help reference, and given a quick

introduction to what he could do with the touch controls. Other users were able to follow the demonstration from the projector that was still connected to the tablet. The controls that were demonstrated are as follows:

- All navigation tools: Orbit, Pan, Walk and Turn
- Area selection
- Creating and activating views
- Focusing on a part by double-tapping on it
- Fitting entire model to view by double-tapping on background
- Using the stylus on a single task to show it to the participants

The users were presented with three exploratory tasks for finding out how the zooming joystick and the measurement and marking tools work. We wanted to find out what they were expecting the tools to do, and did the controls fulfil their expectations. It was also interesting to see how learnable and easy to use the controls were.

Using the stylus was presented in one of the tasks presented by the instructor. This was done to introduce the stylus usage to the participants and then allowing them to explore it, and choose which input method they wanted to use. We were interested in their views of which method is more suitable for them.

5.2.2. Inspection visit

When on the construction site, the evaluation team had an interest to observe certain aspects of the software being in the field use. We were interested in:

- What navigation model the users prefer in the field; touching with fingers or using the stylus?
- What are the user's preferences on the navigation modes and multi-touch gestures?
- Do the users prefer gestures instead of the modes or the buttons or vice versa?
- How much do they use free navigation? Or do they only rely on switching between the views?

- What problems do the users have with the navigation that they chose to use?
- Will the users want to have the zooming glass feature on other input methods than touching? Or using zooming otherwise than in a small area?
- How will they measure objects that are very distant in relationship to each other?
- How will the users access the part details if they need them? Will they find out what they are looking for easily?
- How do the users expect the zooming joystick to behave?

In Figure 5.1 the evaluation team and the subject are shown in the environment of the construction site where the discussion was ongoing. From left to right there are the instructor, the observer, the main test participant and the cameraman.



Figure 5.1: Visiting the construction site.

As part of the evaluation, it was interesting to observe how the users of the tablet device would react to workers coming to ask for certain measurements from him and have him provide the figures on the spot. To simulate this, User #1 was given a task to provide some arbitrary and interesting measurement from the model by pointing out the asked measure in the building itself. This task was not described beforehand to the user and it was given during the visiting round on the construction site.

5.2.3. Feedback for the participants

After the inspection visit was done the group moved back to the construction office for a while to gather feedback from the visit. During this the participants were first given the SUS forms (in [Appendix A](#)) to fill out. After they had filled these they were asked to give feedback of the visit and how they felt about the evaluation so far. This was an important step in debriefing the users and giving them an opportunity to be heard and ask questions of the product if necessary. All proposals and questions were written down and reviewed later as additional material.

5.2.4. Debriefing the user testing

Instant feedback with the evaluation team after the session had been finished was collected as free form discussion and the observations were also recorded with a voice recorder and by taking notes. The significance of these notes was the intuition and immediate observations that the team would otherwise easily forget as more time would pass.

5.3 Analysis of the material

The main method of analysis was the affinity diagramming and the discussions related to that, as described in the background. The setup for the affinity diagramming session consisted of a collection of beforehand written notes, which signified observations from the problems at hand. Those notes were then grouped on a wall according to their affinity to each other. If the notes are not too overlapping in content, they should usually form distinct groupings with reasoning within the team how their affinity was formed.

The analysis of the material began by reviewing the video material gathered from the session. During this review notes of important findings or observations should be written down for further analysis. The notes are supposed to bite size chunks so they describe in as few words as possible the problem or the observation and the writer of the note may elaborate more on demand about the note.

All the hand written notes should also be processed in a similar manner to the video material to produce a number of separated notes from the hand written notes and the instant feedback gathered from the team immediately after the session with the users.

When creating the affinity diagram, the team that was present in the evaluation was also supposed to be present. More people were invited along to speed up the process and share the information. The group that was present in creating the affinity diagram was a mixed group of people from the evaluation team and from the implementation team and everyone's opinions were valued equally.

The participants took turns to place a note onto the wall and explaining their choice as they made it. The distance of the note, in relationship to other notes, signified its affinity to them. Explaining the reasoning of the placement was an important step in sharing the understanding of the meaning placed onto the note, rather than the position of the note. Notes on the wall were allowed to be reorganized as pleased when placing another note, provided that the action was also explained to the other participants. Figure 5.2 displays an example from the work process.



Figure 5.2: Team creating the affinity diagram.

After all the notes prepared from the collected material had been placed onto the wall and some groupings had emerged, the team reviewed the wall and the merged groups. They then tried to value the groups by ordering their importance as findings. The groups were described in text and photographed for later use. The final result is visible in [Appendix D](#).

5.4 Findings from the evaluation

5.4.1. Touch interface was easy to pick up and accepted

All users were able to use the touch interface immediately without any prior briefing. When the test unit was set up and the user was asked to introduce the model to the testing team, they immediately started using the touch interface on the device. Only after specifically asked to do the introduction using the mouse, did the user abandon using touch until further notice.

The users also stated that they preferred using the touch interface instead of the stylus, even regarding the regulations that may enforce the use of protective gloves on site. They said that if anyone had ever used any touch interface previously, then getting to know this touch interface would be easy enough.

5.4.2. Mental model of navigation

We found that the user #1 had a mental model of navigation based on using the physical buttons of the mouse, and that model was migrated to the touch-based user interface directly. The mental model allowed user #1 to use the left mouse button for clicking and dragging, scroll wheel and middle mouse button clicking and dragging. The right mouse button was used only for opening the context menu. We attempted to show how to improve this model without encouragement, but the user did not take it into use.

The users also preferred using the orbiting tool, which had been the default tool in the previously published version of the software. User #1 explored the other tools as well but did not adopt them after trial. Further investigation would be in order to find out why. We assumed that the user was most confident in using the orbiting tool and this is why he, maybe even subconsciously, chose to use only this tool even if it leads him to error situations, such as navigating into a wall.

In addition to just navigating, the users acknowledged that they might start using the snapshots as a safe exit from a difficult position in order

to get back to a familiar position in the model. The users did not use the undo functionality at all. It remained unknown whether the functionality was known or not, and this presented an interesting speculation as to why the users wanted to have a way of easy exit, when was it already there. This in turn proved that either they did not know of the undo feature, or that they knew it, but it had not fulfilled their expectations.

5.4.3. Orbiting tool lead users into error situations easily

The navigational model of user #1 made it so, that even when he was navigating inside the building he was using the orbiting tool. It often got him lost inside the model when some element blocked the line of sight in relation to the orbiting point. The orbiting tool picks the rotation point on the closest surface under the cursor, when the behaviour begins. If that rotation point is far away when the view is inside a building, the speed of movement may be too great for subtly allowing the user to switch the view angle. This could be corrected by adjusting the rate of change either directly or by adjusting the logic by which the rotation point is picked.

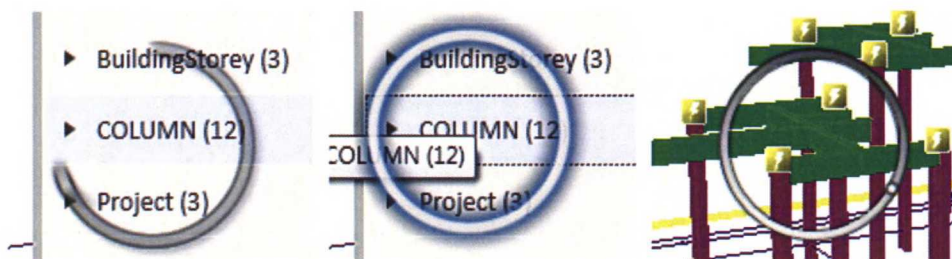
When using the orbiting tool the users tended to end up inside another element blocking the view partly or completely. The users then opened the context menu using the right mouse button to access the hide command for that element. This behaviour suggests that it could be beneficial, when orbiting, to adjust the visibility of the elements that get in the direct line of sight between the view and the rotation point.

5.4.4. View angle is very narrow inside buildings

It remains a speculation, but it seemed that the view angle was too narrow to allow users to see enough when navigating inside buildings. This observation was made when users compared Tekla BIMsight to a product called BIMx, which featured a much wider view angle of the model being presented.

5.4.5. Difficulty of discovering the Tap&Hold gesture

The users had significant trouble in finding out how to make the context menu appear with the touch interface. All three users first attempted the Tap&Hold gesture but they failed to hold for a period long enough and thus the gesture recognition failed. They then attempted other gestures with more fingers and double tapping. Only after instructions did they find the Tap&Hold and were able to successfully open the context menu. After analysing this finding, it was found that the product inhibits the classic usability error suggested by Jakob Nielsen that *“the system should always keep users informed about what is going on”* [56]. We found that the product did not give enough indication that the recognition process had been started. In Windows this is accomplished by showing the user a round circle that fills up during the duration of the Tap&Hold gesture and the user has the needed feedback, as presented in Figure 5.3 and 5.4. The 3D component could not accommodate this response because of the implementation, and the problem was addressed by adding a custom rotating disc to indicate background processing, as presented in Figure 5.5.



Figures 5.3, 5.4 & 5.5: Press and wait indicators in Microsoft Windows 7 and in Tekla BIMsight. The two left images illustrate the animation in Windows 7. Tekla BIMsight has a rotating disc.

5.4.6. Accessing part details was difficult

Users stated in the test, that they wanted to access the part details easily. In the office they attempted to access those details by several means. First they attempted double tapping on parts, which only brought the parts into view for them. The second attempt was finding the details on the context menu, where only the name of the part was

visible. Upon further instruction of user #1 they discovered, that pressing on the name opens the part details on the right-hand panel. It is noteworthy that in the main product of Tekla Building & Construction, Tekla Structures, part details are opened by double clicking on the parts. They also proposed that opening the part details could be done when the part selection changes.

5.4.7. Measurement tool did not match mental model

The measurement tool was implemented to display the distance between two points that the user picks from the model. Since touch interface is a direct action interface, as explained in the background, the tool was specified to require two separate points to be picked by the user. Upon picking each point the user may linger on the point to reveal a magnifying glass to pinpoint the exact position they want to pick. When the finger is released, the point is accepted.

User #1 attempted first to tap certain parts of the model when using the measurement tool for the first time. It seemed as if he was expecting a confirmatory action to follow the first action, and then when nothing happened tried it again. After this strategy failed to yield any satisfactory results he attempted to draw a line from one point to another, presumably to literally “draw a line”. This next model of behaviour was closer to the model the implementation relies on and on an occasion revealed the zooming glass feature to the user very briefly. However, user expected the line to be drawn from the first point to the last point immediately and not that he would be picking the points separate from one another. The user also commented that hitting precise targets was very difficult with the tool. This was also the reason that the user had difficulties in finding the magnifying glass and understanding its behaviour in relation to his actions.

During the visit the team asked the user to measure the installation height of a wire bed and the user complied with the task. After completing the task the user stated that this use case was flawed, because he would always have the information available through the part details embedded in the model file. This lead to the finding that

one of the main use cases for the measurement tool was assumed incorrectly and strongly backed by the user stating: *“This is the kind of information I have at hand on the part details already.”*

5.4.8. Reception of the zooming controls

Users discovered the zooming joystick after being informed of its existence by the instructor. Users then discovered that they can press on the zooming buttons and the view changes accordingly. The draggable joystick control however seemed to puzzle User #1 at least. He could not affect perceptible changes in the 3D view even though he was using the tool as designed. Figure 5.6 displays the movement that the user performed, highlighted with green arrows, and we can clearly see that the result does not indicate any change. The control would have allowed continuing dragging over the bounds of the control to affect a greater movement, and to use the confines of the control to affect subtle zooming.

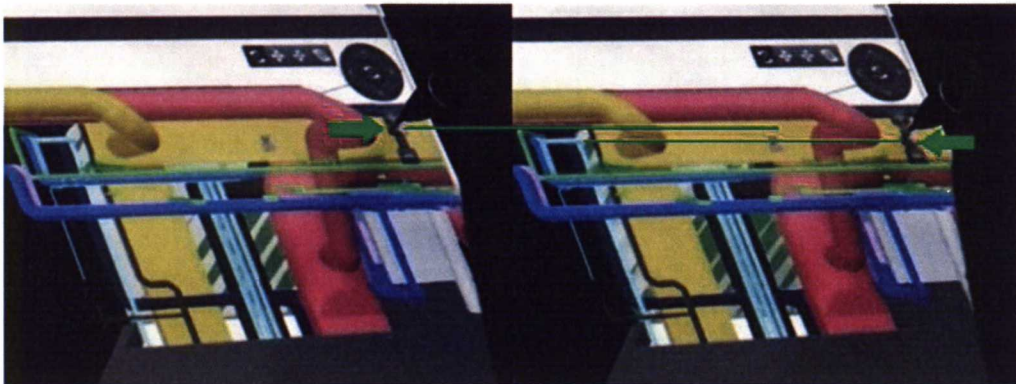


Figure 5.6: User attempted to use the zooming joystick as the visual clue suggests without achieving perceptible results.

On a later discussion after the analysis, the joystick control was explained by the designer to have been implemented to include the dragging control against original design just because it happened to have fit the space nicely. Later on the control was modified to better fit the dragging control and this combination of modifications to the design lead to an inferior design.

The zooming slider should reflect the amount of zoom, instead of just sitting at the center of the slider. It should reflect some absolute

amount of zoom and the slider should be longer to allow finer control over the zoom for the user.

5.4.9. Other observations

Other unexpected features that were observed during the planning process of the usability evaluation are listed below. Most of these problems have already been fixed in the actual software, but they were found during the conducted evaluation. The most critical limitations that were observed in the software were the following:

- Users attempted to open the colouring dialog using touch and found that it did not close automatically if they touched somewhere else in the user interface.
- Two-fingered panning and pinching gestures do not work when the measuring tool is active.
- The navigation circle that allows users to continuously use the tool in use does not indicate the active tool. It should perhaps repeat the information in the center of the navigation circle like mouse cursor reflects the active tool.
- Converting information that comes from the parts does not work correctly if there is a plus or minus sign in front of the value.

It was an observation on behalf of the evaluators that users might benefit from using the hard keys found on the target tablet device. Therefore, a feature request was recorded from this that:

“Add more shortcuts that may be bound to the hard keys found in the target devices sides. There are altogether four shortcut buttons on the device and a directional pad that could be utilized for navigation or other functions, if there were more shortcuts available in the software.”

5.5 Findings from the affinity diagramming

The raw findings listed herein are interpreted from the material analysis with the affinity diagram and presented as they were interpreted from there. The affinity diagram that was created may be found in the [Appendix D](#). Findings are as follows:

1. Touch input was easy to pick up.
2. Hitting small targets was very difficult with touch, because it was not precise enough and too sensitive.
3. Part details are important to the users and information retrieval should be improved.
4. Measurement tool was difficult to use when measuring pipelines from the model. This was a specific use case.
5. Most important measurements were in the part details already. Measuring tool needed more suitable use cases.
6. Users did not know keyboard shortcuts.
7. Users did not use undo.
8. Users preferred using clipping planes instead of hiding parts.
9. Users wanted to have the relevant objects visible and would go to great lengths in hiding obstructing parts.
10. Users preferred using touch instead of the stylus and multi-touch instead of the activated tools.
11. iPad presented difficult competition to the application, because users kept comparing the touch interface to it.
12. Users wished to have access to their documents. They suggested adding links to the objects themselves.
13. BIM on site was seen as a valuable asset that users would like to utilise even more.

Furthermore several problems were recorded from the software, that that were reported to the development team.

1. Clipping planes broke the graphics display.
2. Clipping planes were not cutting the model in straight angle.
3. There were problems when loading models, especially the architecture model was not displayed.
4. View angle was very narrow for navigating inside buildings.
5. Empty notes could not be saved, and single snapshots could not be shared.

5.6 Findings from system usability scale

System usability scale introduced in 2.7.2 was processed from the forms (in [Appendix A](#)) that the users filled out at the end of the evaluation session. As explained before, the results were summed up and multiplied by 2,5 points to reach the overall score which ranges from 25, the worst score possible, to 100, the best score possible. Below are the final results for the SUS questionnaire:

User #1	47,5
User #2	25
User #3	67,5
Summary SUS score	46,7
Percentile of the study	3,50%

As previously stated the SUS scores are not linear in the grading and the percentile the questionnaire scored is more important. The empirical study on SUS indicated that individual evaluations on the SUS score rarely go under 30 points and that the lowest quarter scored under 62.26 points [59]. This means that the evaluation of this system scored in the lowest quarter and that significant improvements on usability are warranted. Figure 5.7 is a more detailed graphical representation of the results. First different user results are presented as how much they contributed points in each question and then the average in the bottom. As the individual scores above already present, User #2 was very critical about the system, User #1 less critical and User #3 almost positive about the system.

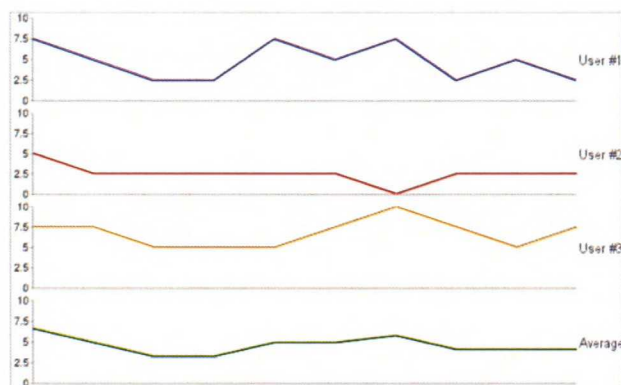


Figure 5.7: Detailed drill down of the SUS results.

5.7 Discussion on the usability test

In relationship to the original plan the test was a success and most of the research questions set before the user testing were answered in observations and in findings. The execution of the test did not follow the plan precisely and there is room for improvement in the test tasks. Should the test have been more formal and structured than what it was, it may have also affected the participants' willingness to give information in such a rich manner as they did. Additionally the usability evaluation footage was composed into a 15 minute annotated video clip containing the evidence from the findings.

The questions that remained without observations were:

- Will the user want to have the zooming glass feature on other input methods than touching? Or using zooming otherwise than in a small area?
- How will they measure objects that are very distant in relationship to each other?

The users stated that they preferred the touch interface instead of the stylus, but it should be further investigated if this view is shared among other users as well and not limited to just this group. Users also preferred using the multi-touch gestures instead of the tool palette and single touch gestures and the four-way navigation controls.

The users seemed to be tapping on the interface a lot and it is worth considering that the user interface should support this kind of activity more. Another approach might be to further study other touch-based user interfaces, in order to understand their fundamental operating logic better and consider mimicking it to a degree.

After additional analysis it was concluded that the mental model chosen for the measuring tool may be wrong and it should be revised later on. Furthermore, when considering the ten usability heuristics, the system should inform the user what was expected, what was the system status, prevent the error situations and help user recover from error situations [55, 56] all of which were violated.

A future investigation should be initiated to find out if there are some tools in the software that do not solve the problem they are meant to solve in the use of a user, does the tool solve another problem for the user and how well the tool matches to the use case associated to it.

6. Results

In this chapter the results will be summed up. The subjects covered here deal with why the development team chose to change the implementation so drastically after the third prototype, what were the problems in the already attempted solutions, what were the good lessons in the previous versions and how they should be applied later on.

6.1 Summary of prototyping

The development team lacked the prior knowledge about touch input systems. Online material provided a starting point, but proved to be of limited use as very detailed implementation documentation was hard to come by. The strongest support came from the API documentation [2, 72, 73, 76, 77]. Additionally the requirements were not very detailed at the time when implementation was supposed to begin and the main requirement remained for long as follows: *“Create any minimal support for tablet devices so that the software is usable.”* This resulted in more research work for the team in finding out details how to do the implementation.

First concern overall was the performance of the system. The team already had experience with the current 3D technology that presented the possible problems with the performance on low-end devices. A decision was made in rather early phase to go ahead with the implementation in any case and wait for more powerful hardware to compensate for possible low performance now. The main target was still to create the essential support for tablet devices. This was a known risk that was chosen so that the business requirements could be fulfilled. The result was that the risk was mitigated in the end with good engineering effort and devotion.

For a long time there was a false feeling of *“quick win”* and that the *“tablet support”* could be done quickly and cheaply. In the sprint planning meetings for sprints 1, 2 and 3 it was estimated that the feature could be finalized in the respective sprint within several days.

Prototypes were demonstrated with success during Sprint 1 and 2 demo meetings. During the sprints 1 and 2 the implementation effort relied on two persons. The team shares a strong sense of devotion and shared responsibility and therefore at the end of each implementation sprint the outcome is evaluated by the entire team. In the middle of implementing the 3rd prototype the internal quality of the approach was questioned and the team started to evaluate the feasibility of the solution. The results from those sprints were not satisfying the expectations of the team. To correct this entire team was dedicated to finding out a better solution. After that the whole team was involved in finding right architecture for the touch support. Series of planning meetings were held to exchange ideas, synchronize and quickly adjust directions. New approach was chosen and quickly implemented with a backtracking list available of known problems and future steps for architecture. The facts leading to this conclusion were the following [78]:

Meaning of “*tablet support*” was not well defined until late implementation phase. No requirements existed that would have been accepted by both the business owner and the development team. Approximately three sprints were spent on prototyping and playing around with different solutions in order to help product owner to decide desired functionality. The only tangible results from this time were the paper prototypes for a renewed UI.

Target hardware devices, namely Motion J3500, were not available for development team until Sprint 3. Combining WPF with native 3D visualization components gave additional technical challenges due to the airspace problems that needed to be resolved. The devices were received in sprint 4 and this enabled the team to start testing the real-life performance too.

Low performance of tablet devices was a known issue. The plan was to give it an attempt to deliver the functionality, even without knowing the real life performance, and wait for better hardware to come available if the problems persists. This risk could not be affected

directly so it could only be mitigated. The result of the 4th prototype fulfilled the internal quality requirements that the team had set. The performance problem was finally solved also as described with the 5th prototype.

6.2 Lessons learned

6.2.1. Prototyping of unknown technology

Lessons learned from the realisation of the risk of an unknown technology were written down as part of the effort of the evaluation described above. These lessons are not unique to this situation but describe the situation very well. To mitigate the risks it was suggested that the following steps should be taken [78]:

1. Do not skip requirements analysis phase. The lack of experience from touch-based user interfaces affected a lot.
2. Invest time into prototyping, research and learning of unknown technology before committing into implementation.
3. Do not start implementation and making long term promises about releasable features when there are many unknowns.
4. Having target devices early is very important to reduce number of surprises.

6.2.2. Indirect interaction is difficult with direct input

Time was spent on investigating how to provide a user interface to accomplish the things that the user should be able to do with the final touch interaction. This was done as an attempt to transform the touch input into mouse events that fed the behaviour mechanism. This attempt was deemed failure on many accounts. It was very difficult to adapt the two mechanisms together, when the basic operation principles are so different in the sense of mouse being an indirect mechanism and touch being direct input. One of the biggest problems with this was the difficulty of adapting the press and release events to mouse events so that the mouse events would be both synchronous and in a proper ordering. This was especially cumbersome when one finger was released after two fingers had been pressed down. Furthermore

the lesson from this mechanism was that it was very difficult and mostly infeasible to attempt to emulate another input device with a different device.

6.2.3. Direct input — experience is critical

Feedback at an early phase is vital for fine tuning a delicate control mechanism. Having the devices readily available helps to gather the first impressions on a new interaction model. One feature where this was constantly apparent, was on the performance of the touch interaction. Many people who were loosely connected to the product development were able to come and try out the current implementation and give immediate feedback with the devices.

It is more natural to have the points under your fingers follow your fingers. It was observed with the 2nd prototype that for example, if you pick two points on a beam and start dragging your fingers across the display you would rather that the points stay under your fingers. It is more intuitive to have the points follow the fingers the same way the fingers are moving. There are individual preferences to this rule however and not everyone might agree with this behaviour. Some might prefer a so called inverted behaviour. In a walk mode this could be so that if finger movements Y-axis move the camera forward on positive axis, it feels unnatural that this control effectively flips around when the camera is looking down. The implementation does not necessarily account for the camera angle change, but now upwards would in the cameras view angle be backwards instead of forwards. User might expect the view to move up when your finger moves up but actually it moves down if this is not accounted for.

In a similar manner actions performed with fingers should be consistent, to allow the user to predict how a control works and not learn the new behaviour. It is a well known and documented usability issue that users should be able to predict how tools behave by having reference in other similar tools.

It could be loosely stated that the “*what you see is what you get*” - principle should be followed as much as possible, in the sense that in a direct input interaction users rely solely upon the direct feedback of the application. By deduction alone users might think, that if the device does not immediately reflect a change upon interaction, then they should maybe try again. This is not always what the designers think is happening, but the mental exercise should prove useful in many an occasion.

6.2.4. Tricks in handling the implementation of touch

Mouse and touch devices behave differently. It was observed that the mouse events were never pushed to the event handler, whereas the touch device seemed to be pushing events to the handler constantly. It is important to make certain that the event handler is not being blocked.

The overlay window was a good solution because it provided a nice division between the types of content that would be made visible to the user. It also acted as a division between the caught input, as the overlay would capture touch events and pass everything else through to the layer below. This in turn could be used in other applications as well to implement a more generic approach to implementing touch controls. A fully generic touch implementation is not likely to exist.

For fluent behaviour with the touch input, it was important that the events were handled in correct order. Implementing actions with events that did not fire in predictable order was very difficult. Filtering was also necessary in this solution while still retaining all the resolution in the events if necessary. The message pump solution was very useful for this purpose and it allowed the developers to choose if the specific action required all the details prior to the action or just the latest event. The critical choice was the level of filtering and where in the processing pipeline would the filtering take place.

One problem that was observed after implementation had already been finished, was that the mark-up tool that is used to paint lines over the 3D view produced smooth lines on the target device, whereas the stylus did not produce so smooth lines. The problem has since been fixed.

Using dot product in gesture recognition worked very well for the needs there were. It yields results quickly and is easy to compute. There are, however, limitations to where it works properly. Extending the gesture recognition or switching to another solution may be in future steps for the development of the software.

One of the future development targets that were not explored at all was to make all the behaviours so that the geometric points underneath user's fingers would be utilised in the behaviours. This could possibly result in more natural user experience as the fingertips would in a way be touching the virtual model.

6.2.5. Rendering performance

The rendering performance was eventually fixed by modifying the calls to the rendering engine so that it would be called only when needed instead of clogging it with requests to render.

Fixing the rendering pipeline problem may have enabled us to utilize the standard Manipulation events that the WPF has built-in. We observed that the RPC implementation had some problems, but maybe we could have been able to cope with it and possibly save some time in the later prototypes. All the effort to fix the handling of the touch events was not in vain but misdirected as a primary focus point. The expected performance was the same fluent movements that were experienced with mouse throughout the implementation period.

However resolving this specific problem was difficult and took a lot of effort altogether to find. We should learn from this that the visible problems seldom are the only ones.

6.2.6. Unknown safe exits in navigation

Users preferred navigating in the model using the orbiting tool. When the users got lost inside the model they did not use easy exits to return to their previous position. Users did not use the undo functionality at all. It remained unknown whether the functionality was known or not, and this presents an interesting speculation as to why the users want to have a way of easy exit, when it already is there. Users acknowledged that there might be times when a safe exit would be useful and that they might use stored views for that. Another question is that did the users associate undo action to navigation, since normally it does not affect it. This should be further studied.

7. Conclusions

Research questions for this study were set to be academic in nature and the intention was to provide a context in which to study the migration of a specialized user interface to incorporate a new input device. As a case study Tekla BIMsight was to be the application context in which to study the research questions. The research goals were set to assist in scoping the subjective work for the study in order to have clear goals on the outcome of this work.

In the background, research methods for evaluating the usability of a specialized user interface were studied at length but the work was found to be very academic or very domain specific. It is difficult to say that there would be common tools for evaluating this and it is the professional skills of the usability engineers to tell if some solution is good or not.

At a very late stage when the background research had almost been completed new information surfaced about the cognitive and motor load measurements. The idea being, that if these load measures were easy enough to calculate without user testing, they could be used to prove or disprove changes in the user interface. Theory for such a method was not located and therefore not discussed further on in the background. This will be subject to future study in the course of further development of the software.

The second research question was intended to drive the study to find out a generic approach to solving the problem of migrating user interface from one input device to support other input devices. Microsoft suggests that the user interfaces should be designed from ground up to support all the intended input devices [73]. Others [19, 21, 47, 49, 51, 50] discussed the problems that need to be addressed when implementing touch controls. These problems were presented in the background research in with collection of otherwise interesting studies [1, 22, 46, 48, 51]. The conclusion is that with each new context

for a user interface the problems need to be resolved again and only experience from such solutions can help alleviate this task.

Specifically designing a user interface to later support other input apparatus is also a bit wasteful, as one might end up designing functionality that will never be used. This also points to the other conclusion, that if the user interface needs drastic changes to support other usage scenarios then it should be re-evaluated and designed to suit the new requirements. Resolving the scope of the work is part of this and it will depend on the available resources.

The third research question was a variation on the first one geared towards the study of touch-based user interfaces. In background research a methods derived from Fitts law [53] was encountered [1, 50], which enabled measuring user performance in selection based tasks and trajectory based tasks i.e. dragging an item on screen. The method was to be used in evaluation of the measurement tool in the software to compare the touch input performance with the same tool used with a mouse. This study was abandoned after the usability evaluation, because critical usability problems were found in the measurement tool. They already displayed that the design of that feature was flawed and it would not bring any valuable insight to test it.

In addition to the research questions, there were a few research goals to refine the scope of this study. The research goals were subjective in nature and intended to provide targeting of the outcomes of the study for the case study.

During the course of this work several problems have been highlighted that were impeding the *“efficient and effective use on a range of devices including desktop and laptop”*. Not all the problems were critical, but many of them were identified as very cumbersome for the user that would affect the positive user experience of the software.

The implementation work done for the specification and implementation of the touch input mechanism were documented as

the case study in this work. This documentation reflects on the problems encountered in the design and implementation of the most relevant problems and did not cover all the possible problems encountered during that period. The implemented touch support is also documented in this book.

The feedback gathered from the usability evaluation with the SUS forms pointed out that the users were not pleased overall with the end results, even when they repeatedly commended some of the solutions. The end results of SUS questionnaire (in Appendix A) were 46,7 points which sets it according to Bangor et al. [59] to be in the lowest quarter. This means that the result was not much better than the worst results in that study. In future other results from a SUS questionnaire may be reflected against this result. Usability evaluation results and all the material has been made available to the development team.

8. Future Steps

The development team spent time on evaluating the user interface and designing improvement to it, but the lack of resources limited the effort that was possible to invest into this work. According to the conclusions presented before the user interface design should incorporate the requirements that come with the input devices. Future works for the user interface will continue and the findings from this book have already been taken into consideration for future steps.

The rendering pipeline solution found for the 5th prototype earlier in the book is a solution that should be investigated further on. The assumption is that further performance gains could be possible, if the solution is applicable to other scenarios as well, or if the solution may be further improved.

The study of the cognitive and motor load measurement will be continued in order to find a tool for the usability experts working close to the development to proving or disproving changes in the design of the user interface. This would be very beneficial if such drastic changes will be done in future of the development work. The efficient use with touch input should still be studied and how the effective use of the user interface could better be supported on small screen devices. Some of this work was started as part of this study and a presentation for an improved design was presented.

Different ways of preventing user errors should prove useful in many of the navigation tools available in the application. Along the same lines more study on the existing UI would be needed. In this study it was found that users did not know there was an undo button available or they did not use it. As mentioned earlier, it might prove that they did not know of the undo feature or that they knew but it had not fulfilled their expectations. Further study on the utilization of the implemented features would be useful in order to make the unused features more visible or redesign them to support user assumptions.

References

- [1] Forlines, C. et al., *Direct-touch vs. mouse input for tabletop displays*. In CHI '11 Proceedings of the 2011 annual conference on Human factors in computing systems. ACM New York, NY, USA.
- [2] Microsoft, 2011. *Windows User Experience Interaction Guidelines - Touch*. [Online]
<http://msdn.microsoft.com/en-us/library/windows/desktop/cc872774.aspx>
[Accessed January 21, 2012]
- [3] Lunden I. 2012. *Apple of our eye: Gartner predicts 665 million tablets in use by 2016, over 45% of them iPad devices*. In TechCrunch news on April 10th, 2012. [Online]
<http://techcrunch.com/2012/04/10/gartner-tablets-apple-ipad-dominate/>
[Accessed April 27th, 2012]
- [4] Guglielmo C. 2012. *Tablet Sales to Skyrocket in 2012, Apple To Keep Lead*. In Forbes magazine online article on April 10th, 2012. [Online]
<http://www.forbes.com/sites/connieguglielmo/2012/04/10/tablet-sales-to-skyrocket-in-2012-apple-to-keep-lead/>
[Accessed April 27th, 2012]
- [5] Tekla Corporation, 2011. *Tekla BIM software*. [Online]
<http://www.tekla.com/international/products/tekla-structures/Pages/Default.aspx>
[Accessed: September 12 2011.]
- [6] Microsoft. 2002. *Pointer Ballistics for Windows XP*. Whitepaper. [Online]
<http://msdn.microsoft.com/en-us/windows/hardware/gg463319>
[Accessed May 5th, 2012]
- [7] Kelly, A. J., Salcudean, S. E. 1993. *Magicmouse: Tactile and kinesthetic feedback in the human-computer interface using an electromagnetically actuated input/output device*. Technical Paper.
- [8] Chapweske, A. 2003. *The PS/2 Mouse Interface*. [Online]
<http://www.computer-engineering.org/ps2mouse/>
[Accessed May 2nd, 2012]
- [9] Logitech. 2012. *Product overview: Logitech Gaming Mouse G500*. [Online]
<http://www.logitech.com/en-us/mice-pointers/mice/devices/5750>
[Accessed May 3rd, 2012]
- [10] Immersion Corporation. 2012. *Haptics In Use — Mobile Devices*. [Online]
<http://www.immersion.com/markets/mobile/products/index.html>
[Accessed May 5th, 2012]

- [11] Senseg. 2012. Webpage. Senseg – Feel the difference. [Online]
<http://senseg.com/> [Accessed May 14th, 2012]
- [12] Wacom. 2010. *Product features: Bamboo Pen tablet*. [Online]
<http://www.wacom.com.hk/product/bamboo-pen> [Accessed May 4th, 2012]
- [13] N-Trig, 2011. About N-Trig. [Online]
<http://www.n-trig.com/Content.aspx?Page=AboutUs>
[Accessed January 26, 2012]
- [14] N-Trig corporation. 2011. *Press Release: N-trig Multi-touch and Pen Solution Currently Shipping on Leading Enterprise Tablet*. [Online]
<http://www.n-trig.com/Content.aspx?Page=PressReleases&PressReleaseId=753>
[Accessed May 2nd, 2012]
- [15] N-Trig corporation. 2009. *Press Release: N-trig Introduces Enhanced Multi-Touch Functionality to Enterprise Market*. [Online]
<http://www.n-trig.com/Content.aspx?Page=PressReleases&PressReleaseId=415>
[Accessed May 3rd, 2012]
- [16] Synaptics corporation. 2012. *TouchPad product description*. [Online]
<http://www.synaptics.com/solutions/products/touchpad>
[Accessed May 2nd, 2012]
- [17] Synaptics Ltd. Synaptics Gesture Suite™ for TouchPads.
[Online] <http://www.synaptics.com/solutions/technology/touchpad>
[Accessed November 17, 2011].
- [18] Apple corporation. 2012. *Apple Magic Trackpad – Product description*. [Online]
<http://www.apple.com/magictrackpad/> [Accessed May 2nd, 2012]
- [19] Ryall, K., Morris M., Everitt K., Forlines C., Shen C. 2006.
Experiences with and observations of direct-touch tabletops.
Horizontal Interactive Human-Computer Systems. Cambridge, MA,
USA: Mitsubishi Electric Research Laboratories.
- [20] Asus Corporation. 2012. *Eee Pad, Transformer Prime, Technical Specification*. [Online]
<http://eee.asus.com/en/eeepad/transformer-prime/specification/>
[Accessed May 3rd, 2012]
- [21] Ghanam, Y., Wang, X., Maurer, F. 2008. *Utilizing Digital Tabletops in Collocated Agile Planning Meetings*. Agile 2007 Conference.
- [22] Ghanam, Y., Wang, X., Park, S., Maurer, F. 2009. *Using Digital Tabletops to Support Distributed Agile Planning Meetings*. Lecture Notes in Business Information Processing, 2009, volume 31, part 6.
- [23] SMART Technologies. 2012. *Products page*. [Online]
<http://smarttech.com/us/Solutions/Visual+collaboration+solutions/Products>
[Accessed May 3rd, 2012]

- [24] Blindmann, G. 2011. Multi-touch Solution Group. *Multitouch technologies*. [Online]
<http://www.multi-touch-solution.com/en/knowledge-base-en/>
[Accessed May 3rd, 2012]
- [25] Weindorf, P., Anderson, D. & Milne, G., 2004. *Cross-point matrix for infrared touchscreen*. US Patent Application 10/353,645. July 29th, 2004.
- [26] Hewlett-Packard Development Company, L.P. 2012. *HP All-in-One and TouchSmart PCs — HP TouchSmart 520 PC*. [Online]
<http://www.hp.com/united-states/campaigns/touchsmart/touchsmart520.html>
[Accessed May 3rd, 2012]
- [27] Dell Corporation. 2012. *Dell SX2210T Flat Panel Monitor User's Guide*. [Online]
<http://support.dell.com/support/edocs/MONITORS/SX2210T/en/ug/about.htm>
[Accessed May 3rd, 2012]
- [28] Natural User Interfaces Group. 2011. *Touchlib: A Multi-Touch Development Kit*. [Online]
<http://nuigroup.com/touchlib/> [Accessed January 21, 2012]
- [29] Microsoft TechNet. 2007. *Physical Features of a Microsoft Surface Unit*. [Online]
<http://technet.microsoft.com/en-us/library/ee692114.aspx>
[Accessed May 7, 2012]
- [30] Multitouch Ltd. 2012. *Company web page — Multitouch*. [Online]
<http://multitouch.fi/> [Accessed May 7, 2012]
- [31] Microsoft. 2011. *The Power of PixelSense™*. [Online]
<http://www.microsoft.com/surface/en/us/pixelsense.aspx>
[Accessed May 5th, 2012]
- [32] Wacom Company, Ltd. 2011. *Software Developer Support, Frequently Asked Questions*. [Online]
<http://www.wacomeng.com/touch/faq.htm>
[Accessed November 15, 2011]
- [33] Samsung. 2012. *Product description: Samsung SUR40 for Microsoft® Surface®*. [Online]
<http://www.samsunglfd.com/solution/feature.do?modelCd=Surface>
[Accessed May 5th, 2012]

- [34] Microsoft Developer Network. 2012. *Microsoft Surface Design and Development*. [Online]
<http://msdn.microsoft.com/fi-fi/windows/desktop/hh241326.aspx>
[Accessed May 5th, 2012]
- [35] Microsoft Corporation. 2010. *Visual Studio 2010 Products*. [Online]
<http://www.microsoft.com/visualstudio/en-us/products/2010-editions>
[Accessed January 21, 2012]
- [36] Internet Archive, Apple Corporation. 2008. *Apple - MacBook Air - Features*. [Online] <http://web.archive.org/web/20080117053522/http://www.apple.com/macbookair/features.html>
[Accessed January 28, 2012]
- [37] Apple Computing Ltd., 2011. *OSX Lion, Multi-Touch Gestures*. [Online] <http://www.apple.com/macosx/whats-new/gestures.html>
[Accessed January 2, 2011].
- [38] Apple Inc. 2012. *Xcode 4 User Guide: About Xcode*. [Online]
<https://developer.apple.com/library/ios/#documentation/ToolsLanguages/Conceptual/Xcode4UserGuide/Introduction/Introduction.html>
[Accessed January 21, 2012]
- [39] iOS App Library Developer Center
<https://developer.apple.com/library/ios/navigation/>
[Accessed January 21, 2012]
- [40] T-Mobile UK. 2008. *T-Mobile G1 Hits the UK*. Press release. [Online]
<http://web.archive.org/web/20090216151437/http://www.opt-development.co.uk/press-office/release.php?id=242>
[Accessed January 28, 2012]
- [41] Carr G. October 14, 2010. *Unity and uTouch* in Canonical Blog. [Online]
<http://blog.canonical.com/2010/10/> [Accessed January 28, 2012]
- [42] Ubuntu Wiki. 2011. *Multitouch/Ginn* in Ubuntu Wiki. [Online]
<https://wiki.ubuntu.com/Multitouch/Ginn> [Accessed January 28, 2012]
- [43] Ubuntu Documentation Team, 2011. *Multitouch*. [Online]
<https://wiki.ubuntu.com/Multitouch> [Accessed January 2, 2012]
- [44] Buxton B., 2010. *A Touching Story: A Personal Perspective on the History of Touch Interfaces Past and Future*. Society for Information Display (SID) Symposium Digest of Technical Papers.
- [45] Buxton B., 2011. *Multitouch Overview*. [Online]
<http://www.billbuxton.com/multitouchOverview.html>
[Accessed January 21, 2012]

- [46] Wu, M. & Balakrishnan, R., 2003. *Multi-finger and whole hand gestural interaction techniques for multi-user tabletop displays*. In Proceedings of the 16th annual ACM symposium on User interface software and technology. UIST '03. New York, NY, USA.
- [47] Bi, X. et al., 2011. *Magic desk: bringing multi-touch surfaces into desktop work*. In CHI '11 Proceedings of the 2011 annual conference on Human factors in computing systems. ACM New York, NY, USA.
- [48] Morris, M. et al., 2006. *Cooperative gestures: Multi-user gestural interactions for co-located groupware*, CHI '06 Proceedings of the SIGCHI conference on Human Factors in computing systems, ACM.
- [49] Casiez G., Vogel D., Pan Q. and Chaillou C. 2007. RubberEdge: Reducing Clutching by Combining Position and Rate Control with Elastic Feedback. In Proceedings of the 20th annual ACM symposium on User interface software and technology (UIST '07). ACM, New York, NY, USA.
- [50] Accot, J., Zhai, S. 1997. *Beyond Fitts' law: models for trajectory-based HCI tasks*. In Proceedings of the SIGCHI conference on Human factors in computing systems (CHI '97). ACM, New York, NY, USA, 295-302.
- [51] McCallum, D.C. & Irani, P., 2009. *ARC-Pad: absolute+relative cursor positioning for large displays with a mobile touchscreen*. In Proceedings of the 22nd annual ACM symposium on User interface software and technology. UIST '09. New York, NY, USA.
- [52] Isaac, S., Melmon, B. 2011. *Kickstarter project description: TouchFire: The Screen-Top Keyboard for iPad*. [Online]
<http://www.kickstarter.com/projects/740785012/touchfire-the-screen-top-keyboard-for-ipad>
 [Accessed December 31, 2011]
- [53] Fitts, P., 1954. *The information capacity of the human motor system in controlling the amplitude of movement*. Journal of experimental psychology, vol. 47 (issue 6), p.381.
- [54] Nielsen, J. 2005. *Ten Usability Heuristics*. [Online]
http://www.useit.com/papers/heuristic/heuristic_list.html
 [Accessed March 1, 2012]
- [55] Nielsen J. 1994. *Usability Inspection Methods*. In Conference companion on Human factors in computing systems (CHI '94). ACM, New York, NY, USA.
- [56] Nielsen, J. *Usability Engineering*. Morgan Kaufman, 1993.
- [57] Beyer H, Holtzblatt K. *Contextual Design: Defining customer centered systems*. Morgan Kaufmann, 1998.

- [58] Brooke J. *SUS - A Quick and Dirty Usability Scale*. In the book: Jordan P, Thomas B, Weerdmeester B & McClelland I. *Usability Evaluation in Industry*. Taylor & Francis, 1996.
- [59] Bangor A, Kortum P & Miller J. 2008. *An Empirical Evaluation of the System Usability Scale*. *International Journal of Human-Computer Interaction*, 24:6, 574-594.
- [60] Eastman C, Teicholz P, Sacks R, and Liston K. *BIM Handbook: A Guide to Building Information Modeling for Owners, Managers, Designers, Engineers, and Contractors*. John Wiley & Sons, Inc., 2008.
- [61] buildingSMART, 2011. *The official website*. [Online]
<http://buildingsmart-tech.org>
[Accessed: September 12 2011.]
- [61] buildingSMART International, 2011. *All Applications*. [Online]
<http://buildingsmart-tech.org/implementation/implementations>
[Accessed: September 12 2011.]
- [62] International Alliance for Interoperability, 2007. *Industry Foundation Classes IFC2x Edition 3 Technical Corrigendum 1*. [Online]
<http://buildingsmart-tech.org/ifc/IFC2x3/TC1/html/index.htm>
[Accessed: September 12 2011.]
- [63] Tekla Corporation, 2011. *Tekla BIMsight help documentation*. [Online]
<http://www.teklabimsight.com/helpcenter/help.jsp>
[Accessed: September 12 2011.]
- [64] Motion Computing Inc. 2011. *Product Specification: Motion J3500*. [Online]
http://www.motioncomputing.com/resources/J3400/J3500_specs_US.pdf
[Accessed: September 12 2011.]
- [65] Tolvanen O., 2011. *Tekla BIMsight Touch Navigation - Software Specification*. Internal document.
- [66] Hewlett-Packard Development Company, L.P. 2012. *HP All-in-One and TouchSmart PCs — HP TouchSmart 520 PC*. [Online]
<http://h20000.www2.hp.com/bizsupport/TechSupport/Document.jsp?objectID=c02861730>
[Accessed May 3rd, 2012]
- [67] Motion Computing Inc. 2011. *Product Specification: Motion F5v*. [Online]
http://www.motioncomputing.com/resources/F5/F5_Product_Specs_en.pdf
[Accessed May 3rd, 2012]
- [68] Acer Inc. 2012. *Product specification: Acer Iconia Tab W500*. [Online]

- <http://us.acer.com/ac/en/US/content/model/LE.Lo803.047>
[Accessed May 3rd, 2012]
- [69] Samsung. 2012. Product specification: Samsung Slate 7. [Online]
<http://www.samsung.com/us/computer/tablet-pcs/XE700T1A-A03US-specs>
[Accessed May 3rd, 2012]
- [70] Tekla Corporation, 2011. *Tekla BIM software*. [Online]
http://www.teklabimsight.com/helpcenter/helpTopic.jsp?topic=bim_RN_1_3
[Accessed March 1, 2012]
- [71] Tuomiario V., 2011. *Software specification Tekla BIM Platform*.
Internal document.
- [72] Microsoft Developer Network, 2011. *Technology Regions Overview*.
[Online] <http://msdn.microsoft.com/en-us/library/aa970688.aspx>
[Accessed March 1, 2012]
- [73] Microsoft Developer Network, 2009. *Touch Input Architecture Overview*. [Online]
<http://msdn.microsoft.com/en-us/library/dd371413%28v=vs.85%29.aspx>
[Accessed March 1, 2012]
- [74] Tekla Corporation. 2012. *Tekla BIMsight 1.4 Release Notes*. [Online]
http://www.teklabimsight.com/helpcenter/helpTopic.jsp?topic=bim_RN_1_4
[Accessed May 4th, 2012]
- [75] Tekla Corporation. 2012. Press Release. *Tekla BIMsight 1.4 takes BIM to the field with Windows tablets*. [Online]
<http://www.tekla.com/us/about-us/news/pages/teklabimsight1.4.aspx>
[Accessed May 4th, 2012]
- [76] Microsoft Developer Network, 2011. *Windows Touch Gestures Overview*. [Online]
<http://msdn.microsoft.com/en-us/library/dd940543%28VS.85%29.aspx>
[Accessed January 2, 2012].
- [77] Microsoft Developer Network, 2011. *WPF and Win32 Interoperation*.
[Online] <http://msdn.microsoft.com/en-us/library/ms742522.aspx>
[Accessed March 1, 2012]
- [78] Otavin A., 2011. *Why touch support failed*. Internal document.

List of Appendices

- A System Usability Scale Questionnaire, in Finnish
- B Equipment Checklist, in Finnish
- C Material Usage Consent Form, in Finnish
- D Affinity Diagram

Appendix A

System Usability Scale

© Digital Equipment Corporation, 1986.

Testaamani tuote: Tekla BIMsight 1.4.1

	Täysin eri mieltä			Täysin samaa mieltä	
1. Olen sitä mieltä, että voisin käyttää tätä tuotetta säännöllisesti.	1	2	3	4	5
2. Tuote on mielestäni liian monimutkainen.	1	2	3	4	5
3. Tuotetta on mielestäni helppo käyttää.	1	2	3	4	5
4. Mielestäni tuotteen käytön oppiminen vaatii kokeneen käyttäjän opastusta.	1	2	3	4	5
5. Mielestäni tuotteen eri toiminnot ovat liitetty toisiinsa onnistuneesti.	1	2	3	4	5
6. Mielestäni tuotteessa on liikaa epäjohdonmukaisuuksia.	1	2	3	4	5
7. Uskon, että useimmat oppivat käyttämään tuotetta hyvin nopeasti.	1	2	3	4	5
8. Mielestäni tuote on hyvin kömpelö käyttää.	1	2	3	4	5
9. Tunsin oloni hyvin luottavaiseksi tuotetta käyttäessäni.	1	2	3	4	5
10. Mielestäni ennen tuotteen käyttöä pitää opetella paljon uusia asioita.	1	2	3	4	5

Appendix B

Tarviketarkistuslista

- ☐ Checklist
- ☐ Motion J3500
- ☐ Motion telakka
- ☐ Hiiri (langaton tai lyhyt johto)
- ☐ Motion laturi
- ☐ Tekla BIMsight viimeisin versio muistitikulla
- ☐ Käytettävyys videokamera
- ☐ Videokameran laturi ja jatkojohto PS3-pinkasta
- ☐ Kolmijalka
- ☐ Jatkojohto
- ☐ SUS-lomakkeita, noin 6 kpl
- ☐ Materiaalikäyttöoikeuksien luovutuspaperit, noin 6 kpl
- ☐ NDA-sopimukset, noin 6 kpl
- ☐ Tekla BIMsight Help printattuna
- ☐ Muistiinpanovälineet kaikille (lehtiö ja **LYIJY**kynät)
- ☐ Äänitallennin
- ☐ VGA-johto

Appendix C

Permission for recording and use of the usability test data

Product/System: Tekla BIMsight 1.4

Company: Tekla oyj

This usability test is part of a course (T-121.5450 Interaction Design and Evaluation) held at Aalto University School of Science and Technology. The test results will be reported on the course and to the company. The recorded material (video/audio/etc) will be handled anonymously if not otherwise agreed with the test user.

The recordings from the test sessions are useful when the results are communicated to the product development team. Possible problem areas in the product/system are easier to demonstrate to the designers with actual test material.

I give my permission to use the material from the test session for the following purposes:

- ☐ on the aforementioned thesis work
- ☐ to the product development team
- ☐ to managers responsible for product development and design
- ☐ in company presentations where the tested product/system is demonstrated
- ☐ in academic conferences and seminars on usability outside the company

Limitations:

Date:

Signature:

Name:

Appendix D

